VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Karolína Šrůtková

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VÝVOJ APLIKACE DEMONSTRUJÍCÍ ZRANITELNOSTI MOBILNÍCH APLIKACÍ
IMPLEMENTATION OF APPLICATION THAT DEMONSTRATES MOBILE APPLICATION VULNERABILITIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Karolína Šrůtková**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Zdeněk Martinásek, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Bc. Karolína Šrůtková ***ID:*** 194405
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Vývoj aplikace demonstrující zranitelnosti mobilních aplikací

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem práce je návrh a implementace mobilní aplikace pro operační systém Android, která umožní prezentaci zranitelnosti mezi mobilní aplikací a serverem (server poskytuje různé služby). V teoretické části práce nastudujte současný stav problematiky, aktuální zranitelnosti mobilních aplikací a dostupné platformy pro vývoj aplikací pro operační systém Android. V rámci analýzy se zaměřte na zranitelnosti exportované aktivity, chybějící autorizace, neoprávněný přístup k datům (SQL injection, Path Traversal), Broadcast Receiverů, služeb a interního úložiště. V praktické části práce realizujte komplexní návrh aplikace, která musí obsahovat všechny výše zmíněné zranitelnosti. Návrh musí být kompatibilní s obrazem PenterepMail, který představuje stranu serveru. Dále realizujte experimentální pracoviště sloužící k vývoji aplikace (použijte Android studio a emulátor Adnroid). V rámci praktické části práce implementujte návrh a otestujte funkčnost aplikace.

DOPORUČENÁ LITERATURA:

[1] SPREITZENBARTH, Michael, et al. Mobile-sandbox: having a deeper look into android applications. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. 2013. p. 1808-1815.

[2] ANDRUS, Jeremy; NIEH, Jason. Teaching operating systems using android. In: Proceedings of the 43rd ACM technical symposium on Computer Science Education. 2012. p. 613-618.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultant: Roman Kümmel (HACKER Consulting s.r.o.)

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zaměřuje na vývoj mobilní aplikace pro operační systém Android, která slouží k demonstraci zranitelností mobilních aplikací. V teoretické části práce je rozebrána bezpečnost mobilních aplikací a její současný stav včetně popisu největších bezpečnostních rizik mobilních aplikací. Dále je v teoretické části zmíněn obecný vývoj mobilní aplikace pro systém Android. V praktické části je popsán vlastní návrh aplikace, do kterého se řadí analýza zranitelností, návrh základních bloků aplikace a výběr vhodných nástrojů pro implementaci. V části popisující implementaci aplikace je uvedena příprava prostředí, struktura vytvářené aplikace a především její samotná implementace. Poslední část obsahuje ukázku zranitelností aplikace a také výsledek jejího otestování.

KLÍČOVÁ SLOVA

Android, mobilní aplikace, zranitelnost

ABSTRACT

This master thesis is focused on an implementation of application for Android operating system that demonstrates mobile application vulnerabilities. Theoretical part contains security of mobile applications and its current state including a description of the biggest security risks and vulnerabilities. In addition, general development of mobile applications for Android is mentioned. In a practical part of the thesis a custom design of the application is described including vulnerabilities analysis, design of basic application blocks and selection of suitable tools for implementation. The section describing the implementation of the application describes the preparation of the environment, the structure of the created application and especially its implementation. The last part contains an example of implemented application vulnerabilities and also the result of its testing.

KEYWORDS

Android, mobile application, vulnerability

ŠRŮTKOVÁ, Karolína. *Vývoj aplikace demonstrující zranitelnosti mobilních aplikací*. Brno, 2020, 76 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vývoj aplikace demonstrující zranitelnosti mobilních aplikací“ jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu diplomové práce panu Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych ráda poděkovala panu Romanu Kümmelovi za cenné rady a připomínky k práci.

Obsah

Úvod **9**

1 Bezpečnost mobilních aplikací **10**
1.1 Mobilní operační systémy 10
1.2 Organizace OWASP a současný stav bezpečnosti mobilních aplikací a operačních systémů 13
1.3 OWASP Mobile Top 10 14
1.4 OWASP Mobile Security Testing Guide 19

2 Požadavky na vývoj mobilní aplikace pro systém Android **23**
2.1 Architektura systému 23
2.2 Prostředí, nástroje a programovací jazyky 25
2.3 Základní bloky aplikace 27

3 Vlastní návrh aplikace **31**
3.1 Analýza zranitelností 31
3.2 Návrh základních bloků aplikace 34
3.3 Výběr vhodných nástrojů a technologií 38

4 Vlastní implementace mobilní aplikace **40**
4.1 Příprava prostředí 40
4.2 Struktura aplikace 41
4.3 Použitá API volání 43
4.4 Implementace jednotlivých aktivit a tříd 43
4.5 Testování aplikace 58

Závěr **63**

Literatura **64**

Seznam symbolů, veličin a zkratek **72**

A Návod ke spuštění aplikace **74**

B Obsah elektronické přílohy **76**

Seznam obrázků

1.1 Propojení všech tří částí OWASP Mobile Security Testing Guide. . . 20
1.2 Ukázka kontrolního seznamu. 22
2.1 Vývojový cyklus mobilní aplikace. 24
2.2 Architektura operačního systému Android. 25
2.3 Základní bloky mobilní aplikace pro systém Android. 28
3.1 Blokový návrh komunikace. 34
3.2 Původní návrh obrazovky před přihlášením. 35
3.3 Původní návrh obrazovky po přihlášení. 35
3.4 Aktivační obrazovka 36
3.5 Přihlašovací obrazovka 36
3.6 Obrazovka s doručenou poštou 37
3.7 Obrazovka s hlavní nabídkou 37
3.8 Obrazovka s detailem zprávy 39
3.9 Obrazovka s kontakty 39
4.1 Běžící virtuální stroj s přidělenou IP adresou. 40
4.2 Běžící webová aplikace na dané IP adrese. 41
4.3 Vývojové prostředí Android Studio s emulátorem. 42
4.4 Ukázka odposlechu broadcast receiveru. 59
4.5 Zaslaná SMS získaná odposlechem. 60
4.6 Kontakty přihlášeného uživatele. 61
4.7 Využití zranitelnosti SQL Injection. 61
4.8 Využití zranitelnosti Path Traversal. 61
4.9 Ukázka zaslání dat dané akci. 61
4.10 Zobrazené oznámení pomocí zavolání dané akce. 62
A.1 Ikona AVD Managera v Android Studiu. 74
A.2 Zjištění portu emulátoru. 75

Seznam tabulek

1.1 Struktura reálného záznamu. 22
2.1 Přehled Android a API verzí. 27
4.1 API pro přihlášení do aplikace 44
4.2 API pro odhlášení se z aplikace 44
4.3 API pro načtení doručených zpráv konkrétního uživatele 45
4.4 API pro načtení odeslaných zpráv konkrétního uživatele 45
4.5 API pro načtení obsahu přílohy 46
4.6 API pro odeslání nové zprávy 46
4.7 API pro přidání nového kontaktu 47
4.8 API pro načtení kontaktů 47
4.9 API pro načtení profilového obrázku 48
4.10 API pro odstranění zprávy 48
4.11 API pro odstranění kontaktu 49

Úvod

Mobilní zařízení se stala v poslední době nedílnou součástí každého z nás, a proto musí být kladen stále větší důraz na jejich bezpečnost. Prvky bezpečnosti by měly být implementovány jak v mobilních operačních systémech, tak v mobilních aplikacích. Útočníci se snaží zneužít zranitelnosti mobilních zařízení nejvíce prostřednictvím škodlivého kódu (Malware) nebo nevyžádané reklamy (Adware). V roce 2020 se jedná například o malware LeifAccess, který zneužívá možnost jediného přihlášení (Single Sign-On) a přístupových služeb k vytvoření účtů, stahování aplikací a zveřejňování falešných recenzí [1].

Hlavním cílem diplomové práce je vlastní komplexní návrh a implementace mobilní aplikace pro operační systém Android, která bude sloužit k demonstraci zranitelností nejen samotné mobilní aplikace, ale také k demonstraci zranitelností mezi ní a serverem poskytujícím různé druhy služeb. Součástí práce je také analýza současného stavu bezpečnosti operačních systémů a mobilních aplikací a analýza konkrétních mobilních zranitelností.

Vytvořená mobilní aplikace by měla přispět k tvorbě bezpečnějších aplikací, především díky možnosti průzkumu zranitelností v reálném čase a reálném prostředí. Aplikace by měla demonstrovat viditelné dopady špatné implementace a měla by tak být jednou z příčin k lepšímu pochopení podstaty zranitelností.

Dílčím cílem práce je analýza a popis bezpečnosti mobilních aplikací. V práci jsou představeny a srovnány nejpoužívanější mobilní operační systémy. Dále je zde rozebrán současný stav bezpečnosti včetně organizace zajišťující bezpečnost mobilních aplikací. Další části práce se věnují mobilním aplikacím a jejich deseti největším bezpečnostním rizikům a metodologii pro testování jejich bezpečnosti.

Dalším cílem byl popis obecného vývoje mobilní aplikace pro systém Android. Konkrétně je v této práci popsán vývojový cyklus mobilních aplikací, architektura systému Android, dostupná vývojová prostředí, nástroje a programovací jazyky a v neposlední řadě také základní bloky mobilní aplikace.

Vlastní přínos práce spočívá v analýze konkrétních zranitelností mobilních aplikací pro systém Android. Na základě této analýzy je vytvořen vlastní návrh mobilní aplikace a zároveň jsou vybrány vhodné nástroje k implementaci. Z návrhu pak vychází vlastní implementace mobilní aplikace, která je detailně popsána. Nakonec je aplikace řádně otestována a je provedena ukázka vybraných implementovaných zranitelností.

1 Bezpečnost mobilních aplikací

Mobilní aplikace je software, který je spustitelný na příručních zařízeních, mezi které patří mobilní telefony, tablety nebo PDA (Personal Digital Assistants). Mezi nejpoužívanější mobilní aplikace se řadí aplikace pro sledování videí a filmů (YouTube, Netflix), hraní her (Pokemon GO, Fruit Ninja), komunikaci na sociálních sítích (WhatsApp, Messenger) nebo čtení zpráv (Twitter, Reddit). Podle analytické společnosti App Annie bylo během roku 2019 celosvětově staženo 204 miliard mobilních aplikací, což značí 45% nárůst oproti roku 2016. Z těchto výsledků lze usuzovat, že se ke zvýšenému počtu stažení úměrně zvýšilo také množství zasílaných dat mezi uživateli a aplikacemi.

Uživatelé posílají svá data aplikacím především za účelem zábavy a poskytnutí služeb. Z poskytnutých dat se tak stávají aktiva uživatelů, protože se jedná o informace, které jsou pro ně cenné a chtějí, aby byly po celou dobu v bezpečí. Bezpečnost je obecně definována jako stav, při kterém ztráty aktiv nepřekročí stanovenou mez. Bezpečnost mobilních aplikací by tak měla být hlavním tématem během celého cyklu vývoje aplikace [2, 3, 4, 5].

1.1 Mobilní operační systémy

Operační systém je typ softwaru, který má za úkol ovládat základní technické prostředky počítače a je zodpovědný za řízení hardwarových částí, za účelem tvorby vhodných podmínek při interakci mezi uživatelem a zařízením. Operační systémy pro mobilní zařízení jsou navrhnuty tak, aby uživateli poskytovaly stejné nebo alespoň podobné funkce, které již dříve byly k dispozici na přenosných počítačích (laptopech). Jedná se například o zasílání emailů, sledování videí, prohlížení novinek nebo hraní her [6, 7, 8].

Mezi nejrozšířenější mobilní operační systémy na trhu patří Android (74%) a iOS (25%). Další minoritní systémy na trhu jsou například KaiOS, Tizen nebo různé mobilní verze operačního systému Windows. Tyto systémy mají podíl na trhu v řádu desetin procenta [9].

Operační systém se také někdy označuje pojmem platforma. Pokud Android a iOS jsou označeny za platformu, většinou je odkazováno na balíček služeb, které na těchto systémech běží. V případě systému Android se může jednat o balíček služeb od firmy Google, jako jsou Gmail, Google Mapy nebo YouTube. U systému iOS se jedná například o iCoud, App Store nebo FaceTime [10].

Android

Android je mobilní operační systém s otevřeným kódem (open-source), který byl na trh uveden v roce 2008 společností Google. Jedná se o operační systém, který je založen na Linuxovém jádru, a jehož hlavní výhodou je široká míra použitelnosti na různých typech zařízení. Android je využíván jako operační systém především v mobilních telefonech, tabletech a chytrých hodinkách. Navíc je považován za hlavní operační systém v zařízeních napojených do tzv. internetu věcí, mezi které se řadí například hlasoví asistenti nebo různé typy senzorů pro řízení teploty či světla.

Oficiálním vývojovým prostředím pro mobilní aplikace na systém Android je Android Studio, ve kterém mohou být aplikace vyvíjeny v programovacích jazycích Java nebo Kotlin. Detailnější popis vývoje, jazyků a prostředí je součástí druhé kapitoly.

Oficiální obchod pro nahrávání a stahování mobilních aplikací pro platformu Android nese název Google Play. Bezpečnost poskytovaných aplikací je zajišťována již při nahrávání aplikace do obchodu. Mobilní aplikace je kontrolována, zda neobsahuje zakázaný obsah (ohrožení dětí, hazardní hry, neschválené látky, atd.), zda nepředstírá jinou identitu, zda implementuje ochranu soukromí včetně ochrany před podvody, zda se nejedná o spam a poskytuje minimální funkčnost, zda neobsahuje malware nebo zda se nejedná o nežádoucí software. Jakýkoliv problém je nahlášen vývojáři, který může daný problém aplikace opravit. Celý tento proces tak poskytuje velmi důkladnou ochranu bezpečnosti, která však nemusí být vždy stoprocentní [6, 11, 12, 13].

V případě, že aplikace není distribuována přes Google Play, ale pomocí jiné třetí strany, má uživatel dvě možnosti. První možností je, že žádaná aplikace může být stažena přes obchod třetí strany, jako je F-Droid nebo Amazon's App Store. Druhá možnost je velmi nebezpečná a spočívá ve stažení aplikace z jakékoliv webové stránky. Před zahájením instalace je uživatel dotázán, zda může být aplikace z neznámého zdroje nainstalována [14, 15, 16].

Na systém Android míří stále větší množství útoků prostřednictvím škodlivých aplikací. V roce 2016 se jednalo o malware Hummingbad, který získával bankovní údaje a obcházel zašifrované emaily. Dále byl detekován například malware s názvem Gooligan, který kradl autentizační tokeny a díky nim získal přístup k datům z aplikací Google Play, Gmail či Google Photos. V roce 2020 se celosvětově stále na předních příčkách umisťuje malware/adware Hiddad, který zobrazuje reklamu skrytým způsobem. V tomto případě je pro uživatele velmi obtížné detekovat zdroj reklamy a popřípadě se jí zbavit [17, 18].

iPhone Operating System (iOS)

Operační systém iOS byl vyvinut společností Apple a na trh uveden v roce 2007. Jeho struktura je odvozena od operačního systému Mac OS X, který je založen na operačním systému UNIX. Na rozdíl od Androidu, iOS je systém s uzavřeným kódem a může být využit pouze v zařízeních od firmy Apple jako jsou iPhone, iPod, iPad nebo Apple Watch.

Pro vývoj aplikací na systému iOS se používá software od firmy Apple s názvem Xcode, který slouží především k editaci kódu, ale také je využíván jako nástroj k odstraňování chyb ve vyvinuté aplikace před vydáním na trh. Mobilní aplikace pro iOS se mohou vyvíjet v programovacích jazycích Swift nebo Objective-C.

Mobilní aplikace pro systém iOS jsou nabízeny prostřednictvím obchodu App Store. Před povolením distribuce vyvinuté aplikace probíhá tzv. „App Review“, během kterého je celá aplikace zrevidována. Aplikace jsou po proběhnutí procesu odmítány na základě velkého množství chyb a pádů aplikace, nefunkčních odkazů, nedostatečného vysvětlení při žádání oprávnění, nekompletních nebo závádějících informací. Celý tento proces kontroly zajišťuje vysokou bezpečnost mobilní aplikace [6, 8, 19, 20, 21].

Systém iOS striktně odmítá jakékoliv aplikace třetích stran pomocí jedinečného blokovacího systému. Nicméně, uživatelé, kteří si nechtějí stahovat aplikace z oficiálního obchodu existují další obchody třetích stran, jako jsou TweakBox, AppValley nebo TutuApp. Stažení těchto aplikací (obchodů) je možné po zamaskování blokovacího systému [22, 23].

Na rozdíl od systému Android na systém iOS nemíří tolik škodlivých aplikací, avšak některé z nich směřují na obě platformy. V roce 2016 se jednalo například o již zmíněný malware Hummingbad nebo Pegasus umožňující ovládat zařízení oběti za účelem zisku citlivých informací. V roce 2020 byla zaznamenána zranitelnost aplikace Mail. Útočníkovi stačilo zaslat zdánlivě prázdnou zprávu uživateli, kterému se po jejím otevření aplikace Mail zastavila a vyžadovala restart zařízení. Během restartování se útočníci mohli dostat k informacím uložených na zařízení [17, 24].

Srovnání systémů Android a iOS

Mezi hlavní výhody systému Android patří jednoduchost hlavního menu a široký výběr možností vzhledu. Uživatelé si tak mohou celý systém nastavit podle sebe, ať už se jedná o vzhled nebo výběr klávesnice, u které si také chválí uživatelsky příjemné rozložení prvků. Systém Android je nabízen jako součást velkého množství mobilních zařízení od různých výrobců, jako jsou Samsung, Huawei či Xiaomi. Tento fakt může být někdy brán jako nevýhoda, protože každé zařízení má v systému

implementovaný jiný styl ovládání. Uživatelé, kteří přecházejí od jednoho zařízení k druhému či k jinému výrobci, nemusí být s novým stylem ovládání ihned spokojeni.

K výhodám systému iOS se rozhodně řadí využití jednotného a praktického rozhraní, které umožňuje stabilní a bezpečné propojení mobilního zařízení s ostatními produkty od firmy Apple. Díky tomu je také na všech zařízeních se systémem iOS stejný styl ovládání. Uživatelé si ale naopak nemohou změnit vzhled zobrazení či klávesnice. Systém iOS je považován za bezpečný z důvodu vydávání častých aktualizací. Hlavní nevýhodou pro uživatele je jednoznačně vysoká pořizovací cena mobilních zařízení s tímto systémem.

Bezpečnost mobilních aplikací je testována jak u obchodu Google Play, tak u obchodu App Store. Přestože má Google Play mnoho bodů, na které se při testování kritérií mobilní aplikace zaměřuje, není ve své politice tak striktní jako je tomu u obchodu App Store.

Obecně lze říci, že každý systém má své výhody a nevýhody. Hodně záleží na tom, jaké mobilní aplikace uživatel často využívá. Například mapy od firmy Google jsou detailnější a uživatelsky lépe navrženy než mapy od firmy Apple. Na druhou stranu, aplikace App Store nabízí svým uživatelům větší množství kvalitnějších aplikací než konkurenční Google Play [25, 26].

1.2 Organizace OWASP a současný stav bezpečnosti mobilních aplikací a operačních systémů

V poslední letech se trend vlastnictví alespoň jednoho chytrého telefonu osobou prudce navýšil. Chytré telefony se tak staly terčem hackerů, kteří se z nich snaží nelegálně získat osobní a citlivá uživatelská data. Stejně jako každý systém, ani mobilní operační systémy včetně mobilních aplikací nejsou bez chyb, což útočníkům nahrává při hledání zranitelností. Využití nalezených zranitelností může mít dopad na systém a samotného uživatele v různém rozsahu. Čím více je mobilní operační systém používán, tím více zranitelností je objevováno.

Mobilní aplikace jsou vždy vyvíjeny pro konkrétní typ operačního systému. Zranitelnosti, které má operační systém, tak mohou být využity právě přes mobilní aplikace. Provázání bezpečnosti jak mobilních aplikací tak operačních systémů je tedy velmi silné a navzájem se doplňuje [27].

V oblasti zajištění především webové bezpečnosti softwaru stojí na předním místě nezisková organizace OWASP (The Open Web Application Security Project). Její hlavní náplní je vydávání článků, sepisování dokumentací, pořádání vzdělávacích kurzů a konferencí. Všechny její činnosti jsou volně dostupné širší veřejnosti, což

přispívá vývojářům a technologům k vytváření velmi bezpečného prostředí pro uživatele [28]. Organizace OWASP je také aktivní v oblasti bezpečnosti mobilních aplikací, v jejímž rámci vytvořila projekt s názvem OWASP Mobile Security Project. V tomto projektu byly zcentralizovány zdroje, které by měly vývojářům a bezpečnostním týmům poskytnout veškeré informace, které potřebují k vývoji a údržbě metod podporujících bezpečnost mobilních aplikací.

Jednou z částí zmíněného projektu je menší projekt s názvem OWASP Mobile Top 10, ve kterém je popsáno deset největších bezpečnostních rizik mobilních aplikací. Přestože byl tento seznam naposledy aktualizován v roce 2016, dodnes je aktuální a využíván řadou vývojářů za účelem vytvoření takové mobilní aplikace, která bude zabezpečená proti všem rizikům zmíněných v seznamu. Mezi Top 10 bezpečnostních rizik patří:

1. Nesprávné používání platformy (Improper Platform Usage).
2. Nezabezpečená datová úložiště (Insecure Data Storage).
3. Nezabezpečená komunikace (Insecure Communication).
4. Nezabezpečená autentizace (Insecure Authentication).
5. Nedostatečná kryptografie (Insufficient Cryptography).
6. Nezabezpečená autorizace (Insecure Authorization).
7. Špatná kvalita kódu (Poor Code Quality).
8. Neoprávněná manipulace s kódem (Code Tampering) .
9. Reverzní inženýrství (Reverse Engineering).
10. Využití vedlejších funkcí (Extraneous Functionality).

Dále je součástí hlavního projektu projekt s názvem OWASP Mobile Security Testing Guide, v němž je popsán bezpečnostní standard a postupy testování bezpečnosti mobilních aplikací [29, 30, 31].

Výše zmíněné dva menší projekty jsou detailněji rozebrány v dalších částech této kapitoly.

1.3 OWASP Mobile Top 10

Struktura kategorií zmíněných v tomto projektu je stejná. Jednotlivé kategorie mají označení M1 až M10 pro snazší odkazování na daná rizika v rámci celé organizace OWASP. Jako první jsou v kategorii zmíněni ve volném překladu agenti hrozeb (Threat Agents). Jedná se o vše, co může využít zranitelnost jako například útočník či malware. Dále jsou zde popsány vektory útoku (Attack Vectors), bezpečnostní slabiny (Security Weakness) a technické a obchodní dopady (Technical/Business Impacts). Poté následují otázky zda je člověk či systém zranitelný v rámci dané kategorie rizik (Am I Vulnerable?) a jak předcházet využití zranitelností (How Do I Prevent)? Na konci kategorie je uveden konkrétní příklad využití zranitelnosti [30].

Nesprávné používání platformy

Tato kategorie pokrývá činnosti, při kterých dochází ke zneužití funkcí nebo využití bezpečnostních ovládacích prvků platforem (OS). Využity mohou být Android intenty (popsány v další kapitole), různá oprávnění nebo funkce TouchID a klíčenka (the Keychain) na systému iOS. Hlavním vektorem útoku je jakékoliv odhalené API (Application Programming Interface) volání. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad vážný.

Klíčenka v systému iOS slouží jako zabezpečené úložiště pro aplikační a systémová data. Běžnou chybou při ukládání dat do klíčenky je využití lokálního úložiště aplikace. Data jsou pak k dispozici v nezašifrované podobě například pro multimediální aplikaci iTunes.

Funkce TouchID může být použita pro oprávněný přístup do mobilní aplikace, avšak nastavení aplikace může být obejito a autentizační proces se tak stává zranitelným [32, 33].

Nezabezpečená datová úložiště

Do této kategorie se řadí činnosti, které vedou k jednoduchému zisku nezabezpečených dat. Pro zneužití zranitelnosti musí útočník mít zařízení fyzicky u sebe nebo oběti zašle malware, který se bude tvářit jako jiná bezpečná aplikace. Výskyt rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad vážný.

Pokud se útočník zmocnil zařízení, může v něm jednoduše prozkoumat všechny soubory a druhy úložišť. Nejvíce nezabezpečených dat se vyskytuje v SQL (Structured Query Language) databázích, v souborech s logy, v XML (eXtensible Markup Language) souborech, v binárních datových úložištích, v úložišti pro cookies nebo na SD (Secure Digital) kartě. Neúmyslný únik dat může být způsoben zranitelnostmi v operačním systému, v použitých knihovnách, v kompilátoru nebo v novém hardwaru [32, 34].

Nezabezpečená komunikace

Mobilní aplikace běžně využívá komunikaci typu klient-server. Zasílaná data musí být přenesena přes telefonního operátora nebo přes internet. Útočníci pak využívají zranitelností na trase mezi zdrojem a příjemcem za účelem zisku přenášených dat. Data mohou být útočníkem získána v rámci lokální sítě přes nezabezpečenou síť, přes router, přes proxy servery nebo s pomocí malwaru. Výskyt rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad vážný.

Mobilní aplikace nejsou ve většině případů navrženy tak, aby zabezpečovaly síťový provoz. Provoz je zabezpečen pouze při autentizaci pomocí protokolů SSL/TLS

(Secure Sockets Layer/Transport Layer Security). Riziko odhalení nebo zisk zasílaných dat je tak velmi vysoké. Největší šanci na zisk dat mají útočníci na nezabezpečené wi-fi síti. OWASP přesto upozorňuje, že riziko nelegálního zisku dat je také v ostatních typech komunikace jako je TCP/IP (Transmission Control Protocol/Internet Protocol), Bluetooth, NFC (Near Field Communication), GSM (Global System for Mobile Communications), 3G (Third Generation), SMS (Short Message Service) a další. K nejběžnějším praktikám útočníků je odposlouchávání komunikace neboli útok mužem uprostřed (Man in the Middle (MiTM)) [32, 35].

Nezabezpečená autentizace

Nezabezpečená autentizace znamená, že zařízení není schopno správně rozeznat, kdo se do aplikace přihlašuje. Zranitelnosti autentizace jsou zneužívány pomocí automatizovaných skriptů a znalost autentizačního schématu, který je využíván v mobilní aplikaci, je pro útočníky nejdůležitější. Jakmile útočník ví, jak schéma funguje a jak ho obejít, může se do aplikace bez problémů přihlásit. Výskyt rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad při využití zranitelnosti je vážný.

U mobilních aplikací je slabší autentizace způsobena využitím speciálního typu vstupu. Tento typ vstupu velmi podporuje krátká hesla ve formě čtyř-číselného PIN (Personal Identification Number) kódu. Využití tohoto vstupu je způsobeno faktem, že uživatelé nemusí být po celou dobu relace online, jako je tomu u webových aplikací. Autentizace v mobilní aplikaci tak může probíhat offline, avšak všechny důsledky způsobené tímto typem vstupu musí vývojáři vzít v úvahu při implementaci.

Mobilní aplikace je zranitelná vůči nezabezpečené autentizaci pokud může anonymně vykonávat API požadavky zasílané ze serveru bez poskytnutí přístupového tokenu neboli elektronického klíče. Dále je aplikace zranitelná pokud ukládá hesla nebo sdílená tajemství do lokálního úložiště nebo pokud je uživateli dovoleno používat slabé heslo [32, 36].

Nedostatečná kryptografie

Pokud je útočník schopen jednoduše rozšifrovat získaná zašifrovaná data, došlo v mobilní aplikaci k použití nedostatečných nebo chybných kryptografických prostředků. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad při využití zranitelnosti je vážný.

Bezpečné šifrování dat v mobilních aplikacích je založeno na certifikátech vydaných certifikační autoritou. Poté, co je ověřen podpis certifikátu, uživatel získá dešifrovaná data. Nicméně, existuje mnoho veřejně dostupných nástrojů, které umožní obejití této metody. U systému iOS je útočník schopen pomocí určitého nástroje

stáhnout aplikaci do zařízení, které nemá v systému žádná omezení. V tomto zařízení je aplikace nainstalována, data dešifrována a je vytvořen snímek stavu systému, tzv. snapshot. Tento snímek s dešifrovanými daty aplikace je vrácen zpět do původní paměti zařízení. Jakmile je aplikace spuštěna uživatelem, útočník ji může zanalyzovat a získat uživatelská data.

Data mohou být snadno dešifrována i v případě, že je použit špatný proces správy klíčů. Tento problém může nastat ve chvíli, kdy vývojář v aplikaci použije bezpečný šifrovací algoritmus, ale špatně ho implementuje. O problém se jedná, pokud jsou zašifrované klíče uloženy do stejného adresáře, který je všem dostupný pro čtení a ve kterém je uložen zašifrovaný obsah, nebo pokud jsou klíče napevno zakódované v binárním souboru.

OWASP také doporučuje, aby vývojáři měli povědomí o šifrovacích algoritmech a protokolech, které již v minulosti byly prolomeny nebo neposkytují dostatečné zabezpečení pro dnešní systémy [32, 37].

Nezabezpečená autorizace

Autorizace je v mnoha případech chybně zaměňována s autentizací. Autentizace je proces, při kterém dochází k identifikaci osoby či uživatele. Naopak během autorizace se kontroluje, zda již autentizovaná osoba má práva k přístupu do žádané aplikace, složky nebo k souboru. Oba dva procesy jsou spolu velmi propojeny, protože po autentizaci by vždy měla následovat autorizace. Obdobně jako u obejití autentizace, je pro útočníka důležité, aby znal autorizační schéma. V případě, že v něm útočník našel zranitelnost, může se do aplikace přihlásit jako legitimní uživatel a poté v ní provádět další akce k zisku dat. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad při využití zranitelnosti je vážný.

Mobilní aplikace může mít nezabezpečenou autorizaci ve chvíli, kdy uživatel může pomocí zadaného vstupu přistupovat k objektům přímo. Tato zranitelnost je známá pod zkratkou IDOR (Insecure Direct Object Reference). Útočník tak může při zneužití zranitelnosti získat přístup k databázi nebo souborům.

Dále je správná autorizace ohrožena, pokud mobilní aplikace zasílá uživatelské role a jejich oprávnění jako součást požadavku na server nebo pokud není autorizace vyžadována u skrytých funkcionalit, které se využívají během vývoje [32, 38].

Špatná kvalita kódu

Problém s kvalitou kódu je běžný u mnoha mobilních aplikací. Vývojáři se dopouští chyb zejména v používání nejednotných programovacích vzorů, v sepisování obtížně čitelné dokumentace nebo ve špatné kontrole velikosti délky dat. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti obtížná a dopad při využití zranitelnosti je

mírný. Přestože jsou chyby v kódu manuálně špatně detekovatelné, útočníci využívají nástroje třetích stran, které umí provést na kódu statickou analýzu. Nástroje dokáží odhalit například únik paměti (memory leak) nebo přetečení bufferu (buffer overflow). Zneužít tyto zranitelnosti není obtížné a útočníkům je díky nim umožněno vkládat cizí kód do kódu mobilní aplikace nebo z kódu pouze získat určité informace [32, 39].

Neoprávněná manipulace s kódem

Žádná mobilní aplikace není imunní vůči neoprávněné manipulaci s kódem, protože aplikační kód vždy běží v prostředí, které není pod kontrolou organizace vyvíjející aplikaci. Toto prostředí se dá ale změnit, což nahrává útočníkům, kteří jsou schopní díky změnám prostředí modifikovat kód aplikace. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad při využití zranitelnosti je vážný.

Ve většině případů je s kódem manipulováno prostřednictvím mobilních aplikací, které jsou k dispozici ke stažení v obchodech třetích stran. Tyto aplikace se tváří jako jejich originální verze, avšak jsou uzpůsobeny tomu, aby po jejich instalaci do konkrétního zařízení byl v nich útočník schopen pozměnit kód, dynamicky měnit obsah paměti, pozměnit systémová API volání nebo modifikovat aplikační data. Těmito způsoby může útočník pozměnit funkce softwaru, což může vést například k peněžnímu zisku. Nejvíce podvodných aplikací se objevuje v kategorii her a správy financí.

Mobilní aplikace je zranitelná, pokud není při běhu programu schopna zachytit přidání nebo změnu kódu. Pro úspěšné zneužití této zranitelnosti si útočník stáhne originální aplikaci, získá z ní binární kód, který následně upraví a vytvoří tak velmi podobnou škodlivou aplikaci. Tu pak nahraje do obchodu s mobilními aplikace a čeká, dokud si ji někdo nestáhne a nespustí [32, 40].

Reverzní inženýrství

Reverzní inženýrství je činnost, při které se útočník pokouší analyzovat legálně získanou mobilní aplikaci ve svém místním prostředí. Pomocí různých nástrojů je útočník schopen získat zdrojový kód, použité knihovny nebo algoritmy. Získané znalosti mohou být dále použity pro odhalení informací na serverech nebo ke krádeži duševního vlastnictví.

Nejvíce zranitelné jsou mobilní aplikace naprogramované v jazycích Java, .NET, Objective-C nebo Swift, protože tyto jazyky umožňují dynamické prohlížení stavu aplikace během jejího používání. Tato zranitelnost se u aplikací objevuje z důvodu jejího ladění, tzv. debugging. Vývojářům je tak umožněno získat spoustu informací o běžících procesech a o chování aplikace. Pokud je debuggování povoleno i ve finální

verzi nabízené aplikace, je velmi pravděpodobné, že se aplikace stane cílem útoku. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti jednoduchá a dopad při využití zranitelnosti je mírný [32, 41].

Využití vedlejších funkcí

Vedlejší funkce jsou využívány vývojáři za účelem získání přesných informací o fungování mobilní aplikace. Jedná se například o funkce pro lepší přistup k backendovým serverům, pro vytváření logů nebo pro tvorbu testovacího prostředí. Tyto funkce nemají dopad na funkčnost vyvíjené aplikace a tudíž není potřeba tyto funkce zahrnovat do výsledné verze aplikace. Nicméně, vývojáři často zapomínají na důležitost odstranění těchto funkčností z finálního kódu. Výskyt tohoto rizika je běžný, zjistitelnost zranitelnosti průměrná a dopad při využití zranitelnosti je vážný.

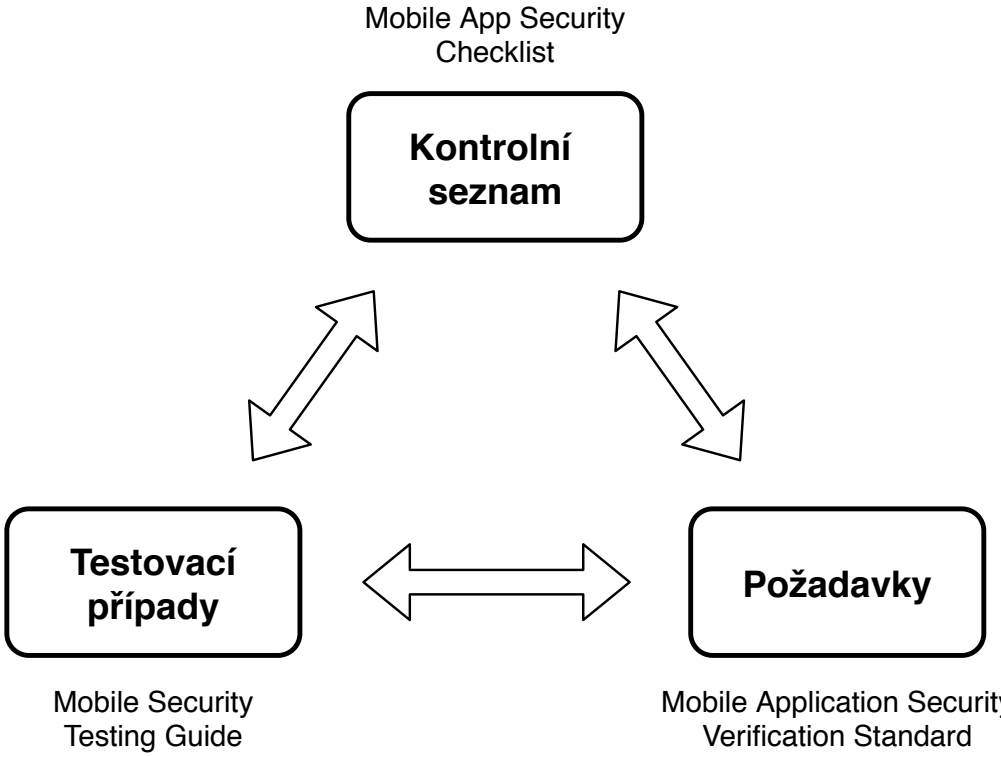
Útočník si nejdříve legálně stáhne originální aplikaci a poté pomocí logovacích a konfiguračních souborů aplikace zkoumá, zda se v ní nevyskytuje nějaká vedlejší tedy pro něj skrytá funkcionalita. Ve většině případů nejsou vedlejší funkcionality útočníkovi k užitku, avšak někdy mu mohou například prozradit, jak fungují backendové servery [32, 42].

1.4 OWASP Mobile Security Testing Guide

OWASP Mobile Security Testing Guide se skládá ze tří hlavních částí. První část nese název **Mobile App Security Verification Standard (MASVS)** a jsou v ní sepsány požadavky na bezpečnost mobilních aplikací. Druhá část s názvem **Mobile Security Testing Guide (MSTG)** obsahuje rozsáhlý návod pro bezpečnostní testování mobilních aplikací běžících na systémech Android a iOS. Třetí část je pojmenována jako **Mobile App Security Checklist**, ve které jsou obsaženy kontrolní seznamy vycházející jak z MSTG tak z MASVS. Propojenost všech tří částí je zobrazena na obr. 1.1. Požadavky uvedené v MASVS by měly být brány v úvahu ve fázi plánování a vytváření architektury mobilní aplikace. MSTG a kontrolní seznamy by naopak měly být využity během a po konci vývoje a při vytváření manuálních nebo automatických testů [31, 43].

Mobile Application Security Verification Standard

MASVS reprezentuje standard pro bezpečnost mobilních aplikací. V MASVS jsou detailně popsány požadavky, které musí být splněny, aby mobilní aplikace byla považována za bezpečnou. V MASVS jsou definovány tři úrovně ověřování - úroveň 1



Obr. 1.1: Propojení všech tří částí OWASP Mobile Security Testing Guide.

(Level 1 - L1), úroveň 2 (Level 2 - L2) a ověření odolnosti vůči reverznímu inženýrství (Resiliency - R) . V úrovni 1 jsou obsaženy obecné požadavky na bezpečnost, zatímco úroveň 2 se více zaměřuje na bezpečnostní požadavky, které by měly splňovat aplikace zacházející s citlivými daty. Ověření odolnosti slouží k dodatečnému zavedení ochranných prvků.

Každá mobilní aplikace by měla splňovat požadavky spadající do úrovně 1. Aplikace by tak měla být ochráněna vůči zneužití běžných zranitelností. Aplikace implementující požadavky úrovně 2 jsou více odolnější vůči sofistikovanějším útokům. Úroveň ověření odolnosti zabraňuje hrozbám na straně klienta, kde konečný uživatel může být považován za útočníka nebo kde je přímo ohrožen operační systém.

Dokument MASVS je rozdělen na dvě části. V první části jsou zdokumentovány bezpečnostní modely, úrovně ověřování a doporučení, jak s tímto standardem zacházet v praxi. Druhá část se věnuje detailnímu popisu bezpečnostních požadavků a přiřazení úrovní ověřování jednotlivým požadavkům. Požadavky jsou rozřazeny do osmi kategorií na základě technického cíle nebo rozsahu [44].

Mobile Security Testing Guide

V MSTG je zdokumentován obsáhlý manuál pro testování požadavků na bezpečnost mobilních aplikací pro systémy Android a iOS. V manuálu jsou popsány procesy a techniky ověřování požadavků vycházejících z MASVS a navíc poskytuje základní informace pro provedení kompletních bezpečnostních testů. Obecně lze říci, že MSTG obsahuje testovací případy, podle kterých by testeři měli postupovat. MSTG se skládá ze tří hlavních částí:

- Obecný průvodce testováním (General Testing Guide).
- Průvodce testováním pro Android (Android Testing Guide).
- Průvodce testováním pro iOS (iOS Testing Guide).

V části *Obecný průvodce testováním* je obsažena metodologie pro testování bezpečnosti a techniky obecné analýzy zranitelností. Dále obsahuje testovací případy, které jsou nezávislé na operačním systému, jako je autentizace, řízení relace, síťová komunikace a kryptografie. Průvodce testováním pro Android i iOS mají totožnou strukturu. V průvodci jsou zahrnuty informace o platformě, základní testování bezpečnosti a testovací případy pro různé kategorie bezpečnostních rizik, které se ve většině případů shodují s riziky zmíněnými v projektu OWASP Mobile Top 10 [43].

Mobile App Security Checklist

Tato část vychází z částí MSTG a MASVS a obsahuje kontrolní seznamy, ve kterých je každý záznam mapován na současnou verzi MSTG a MASVS. Soubor s kontrolními seznamy je vytvořen pomocí tabulky v aplikaci Excel. Součástí souboru jsou kontrolní seznamy pro zajištění požadavků na bezpečnost mobilních aplikací a pro vyhodnocení míry odolnosti aplikací vůči reverznímu inženýrství. Poslední důležitou částí je shrnutí pro management, ve kterém je zobrazen přehled splněných a nesplněných požadavků v podobě tabulky a diagramů. U každého záznamu je uvedeno:

- ID - identifikační číslo,
- (MSTG-ID) - identifikace v rámci MSTG,
- Detailed Verification Requirement - popis požadavku k ověření,
- Level 1 - zahrnuto v úrovni 1,
- Level 2 - zahrnuto v úrovni 2,
- Status - stav testování,
- Testing Procedure - testovací postup,
- Comment - komentář (neuváděn vždy).

Položku *Status* vyplňuje tester a může nabývat hodnot Pass (prošlo), Fail (selhalo) nebo N/A (Not Applicable - požadavek se nevztahuje na aplikaci). V kolonce *Testing Procedure* je uveden klikatelný název kapitoly, který testera přesměruje

přímo na danou kapitolu v MSTG. V tab. 1.1 je zobrazen příklad struktury reálného záznamu [45]. Pro lepší představu je na obr. 1.2 ukázána větší část kontrolního seznamu v aplikaci Excel. Z důvodu velké velikosti tabulky kontrolních seznamů nebylo možné zobrazit všechny její části.

Tab. 1.1: Struktura reálného záznamu.

ID	2.3
MSTG-ID	MSTG-STORAGE-3
Detailed Verification Requirement	Žádná citlivá data nejsou zahrnuta v logách aplikace.
Level 1	Ano
Level 2	Ano
Status	Prošlo
Testing Procedure	Testing Logs for Sensitve Data

ID	MSTG-ID	Detailed Verification Requirement	Level 1	Level 2	Status
V1		**Architecture, design and threat modelling**			
1.1	MSTG-ARCH-1	All app components are identified and known to be needed.	✓	✓	
1.2	MSTG-ARCH-2	Security controls are never enforced only on the client side, but on the respective remote endpoints.	✓	✓	
1.3	MSTG-ARCH-3	A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.	✓	✓	
1.4	MSTG-ARCH-4	Data considered sensitive in the context of the mobile app is clearly identified.	✓	✓	
1.5	MSTG-ARCH-5	All app components are defined in terms of the business functions and/or security functions they provide.		✓	N/A
1.6	MSTG-ARCH-6	A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures.		✓	N/A
1.7	MSTG-ARCH-7	All security controls have a centralized implementation.		✓	N/A
1.8	MSTG-ARCH-8	There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57.		✓	N/A
1.9	MSTG-ARCH-9	A mechanism for enforcing updates of the mobile app exists.		✓	N/A
1.10	MSTG-ARCH-10	Security is addressed within all parts of the software development lifecycle.		✓	N/A
1.11	MSTG-ARCH-11	A responsible disclosure policy is in place and effectively applied.		✓	N/A
1.12	MSTG-ARCH-12	The app should comply with privacy laws and regulations.	✓	✓	
V2		**Data Storage and Privacy**			
2.1	MSTG-STORAGE-1	System credential storage facilities need to be used to store sensitive data, such as PII, user credentials or cryptographic keys.	✓	✓	
2.2	MSTG-STORAGE-2	No sensitive data should be stored outside of the app container or system credential storage facilities.			
2.3	MSTG-STORAGE-3	No sensitive data is written to application logs.	✓	✓	
2.4	MSTG-STORAGE-4	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	✓	✓	
2.5	MSTG-STORAGE-5	The keyboard cache is disabled on text inputs that process sensitive data.	✓	✓	
2.6	MSTG-STORAGE-6	No sensitive data is exposed via IPC mechanisms.	✓	✓	
2.7	MSTG-STORAGE-7	No sensitive data, such as passwords or pins, is exposed through the user interface.	✓	✓	
2.8	MSTG-STORAGE-8	No sensitive data is included in backups generated by the mobile operating system.		✓	N/A
2.9	MSTG-STORAGE-9	The app removes sensitive data from views when moved to the background.		✓	N/A
2.10	MSTG-STORAGE-10	The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.		✓	N/A
2.11	MSTG-STORAGE-11	The app enforces a minimum device-access-security policy, such as requiring the user to set a device passcode.		✓	N/A
2.12	MSTG-STORAGE-12	The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app.		✓	N/A
2.13	MSTG-STORAGE-13	No sensitive data should be stored locally on the mobile device. Instead, data should be retrieved from a remote endpoint when needed and only be kept in memory.		✓	N/A
2.14	MSTG-STORAGE-14	If sensitive data is still required to be stored locally, it should be encrypted using a key derived from hardware backed storage which requires authentication.		✓	N/A
2.15	MSTG-STORAGE-15	The app's local storage should be wiped after an excessive number of failed authentication attempts.		✓	N/A

Obr. 1.2: Ukázka kontrolního seznamu.

2 Požadavky na vývoj mobilní aplikace pro systém Android

Vývoj nejen mobilní aplikace obvykle následuje stejný cyklus, který se skládá z šesti hlavních částí:

1. plánování.
2. Sepsání technické dokumentace.
3. Design.
4. Vývoj.
5. Testování.
6. Zveřejnění a údržba.

Ve fázi plánování probíhá stanovení cílů, analýza požadavků, výběr nástrojů a technologií, popřípadě vytvoření marketingové strategie. Poté během fáze sepsání technické dokumentace dochází k vytvoření dokumentu, ve kterém jsou podrobně popsány postupy pro splnění požadavků. Fáze designu zahrnuje vytvoření vizuální podoby obrazovek. Následuje fáze vývoje, během kterého jsou brány v potaz výstupy všech předešlých fází, na jejichž základě se začne vytvářet finální funkční aplikace. Po vývoji přichází fáze testování neboli kontrola kvality vytvořené aplikace. Pokud je aplikace řádně otestovaná, může být zveřejněna v obchodě s mobilními aplikacemi či předána konkrétnímu zákazníkovi. Vývojářský tým se pak dále stará o údržbu a vytváří nové vlastnosti aplikace.

Znázornění celého cyklus je k dipozici na obr. 2.1. Z obrázku je vidět, že cyklus nekončí poslední fází, ale vše se opět opakuje například z důvodu údržby, změny požadavků nebo vytvoření nové funkce aplikace [46].

2.1 Architektura systému

Jak již bylo popsáno v předešlé kapitole, Android je mobilní operační systém s otevřeným kódem a je založen na Linuxovém jádru. Za účelem vývoje bezpečné mobilní aplikace je nutné znát architekturu systému, která je zobrazena na obr. 2.2. Linuxové jádro (Linux Kernel) je základem operačního systému Android a poskytuje mu klíčové bezpečnostní vlastnosti jako je uživatelský model oprávnění, izolace procesů, rozšířené mechanismy pro bezpečnou meziprocesorovou komunikaci a odstranění zbytečných a potencionálně nebezpečných částí jádra [47].

Hardwarová abstrakční vrstva (Hardware Abstraction Layer) poskytuje standardní rozhraní, která umožňují předávat informace o hardwarových schopnostech zařízení vyšším vrstvám. Tato to vrstva je složena z několika modulů, z nichž každý implementuje rozhraní pro konkrétní typ hardwaru, jako je kamera nebo bluetooth.

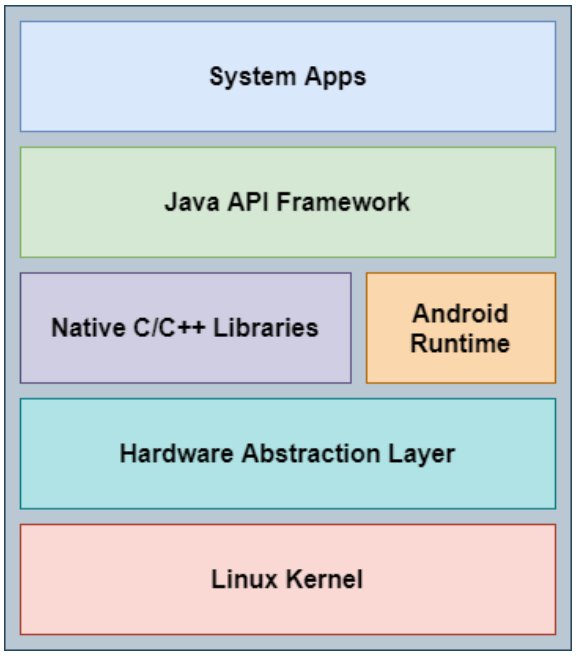
Obr. 2.1: Vývojový cyklus mobilní aplikace.

Android Runtime je část architektury, která má za úkol spouštět několik virtuálních zařízení na zařízeních s nízkou pamětí pomocí DEX (Dalvik Executable) souborů. DEX je formát navržený přímo pro systém Android a je optimalizován tak, aby zanechával minimální paměťovou stopu.

Mnoho základních systémových částí a služeb je vytvořeno pomocí nativních knihoven napsaných v jazycích C a C++. Vrstva nativní C/C++ knihovny (Native C/C++ Libraries) tak poskytuje informace o funkcionalitách některých nativních knihoven přímo API frameworku. Aplikace může například přistupovat ke knihovně OpenGL ES, která jí umožní manipulaci s 2D a 3D grafickými objekty.

Přes Java API Framework je možné přistoupit ke všem sadám funkcí systému Android. Tato API volání vytváří základní bloky potřebné k vytvoření mobilní aplikace pomocí komponent modulárního systému a pomocí služeb. V aplikaci mohou být například implementovány vlastní notifikace a vlastní vzhled nebo jí může být umožněn přístup k jiným aplikacím.

Systémové aplikace (System Apps) jsou mobilní aplikace, které jsou již součástí systému Android. Jedná se například o aplikace Email, Zprávy, Kalendář, Internetový prohlížeč nebo Kontakty. Uživatel však není povinen tyto systémové aplikace využívat, a proto je může nahradit jinými aplikacemi třetí strany [48].



Obr. 2.2: Architektura operačního systému Android.

2.2 Prostředí, nástroje a programovací jazyky

Pro vývoj mobilních aplikací na systém Android existuje celá řada vývojových prostředí, nástrojů a programovacích jazyků. Mezi vývojová prostředí podporující vývoj mobilních aplikací na systém Android jsou Android Studio, Eclipse, Visual Studio a Xamarin, NetBeans, Cordova a další. Oficiálním vývojovým prostředím je Android Studio, avšak použití jiných prostředí není výjimkou a odvíjí se od vývojářova záměru. Mobilní aplikace na systém Android je doporučeno vyvíjet v programovacích jazycích Kotlin nebo Java, ale vývojář může také použít jazyky C/C++, C#, JavaScript nebo Python.

Během vývoje může také nastat situace, kdy vývojové prostředí neposkytuje nutnou funkčnost. Vývojáři tak mohou použít jiné dostupné nástroje, které podporují vývoj a běh mobilních aplikací. Jedná se například o nástroj GameMaker: Studio, které poskytuje možnosti pro tvorbu 2D her, nebo o nástroj Vysor, který umožňuje zrcadlení obrazovky mobilního zařízení na obrazovku počítače [49, 50, 51].

Android Studio

Oficiálním IDE (Integrated Development Environment) neboli vývojovým prostředím pro mobilní aplikace pro systém Android je Android Studio od české společnosti JetBrains. Mezi jeho hlavní vlastnosti patří chytrý editor kódu, který umí našeptávat vývojáři jeho vhodné dokončení. Dále mezi jeho vlastnosti patří editor vizuálního

rozvržení, díky kterému může vývojář rychle a efektivně navrhnout design aplikace pro jakoukoliv velikost displeje. V Android Studiu je také implementován APK (Android Application Package) analyzátor poskytující možnosti inspekce mobilní aplikace. Mnoho vývojářů také oceňuje rychlý emulátor, který je nedílnou součástí Android Studia.

Při vytváření nového projektu si vývojář může vybrat, ve kterém programovacím jazyku se aplikace bude vyvíjet. Na výběr je Java nebo Kotlin, avšak během vývoje může být do projektu také přidána část v jazycích C a C++ [52, 53, 54].

Java

Jazyk Java byl vyvinut společností Sun Microsystems a na trh uveden v roce 1995. Dodnes se řadí mezí nejpopulárnější programovací jazyky, především díky jeho nezávislosti na operačním systému či platformě. Jazyk Java poskytuje vývojářům velké množství knihoven, které jim velmi zjednodušují práci. Jazyk dále umožňuje automatické přidělení a uvolnění paměti nebo souběžnost aplikací. Do roku 2019 byl jazyk Java brán jako hlavní jazyk pro vývoj mobilních aplikací na systém Android [55].

Kotlin

Kotlin byl vyvinut v roce 2011 společností JetBrains jako reakce na chybějící možnosti v jiných jazycích. V roce 2019 se společnost Google rozhodla, že hlavním programovacím jazykem mobilních aplikací na systém Android bude právě Kotlin. Jazyk Kotlin je expresivní a stručný, což umožňuje vývojářům snížit množství napsaného kódu, a tak zvýšit jejich produktivitu. Další jeho výhodou je podpora vývoje bezpečného kódu. Kotlin totiž pomáhá vývojářům, aby se vyhnuli častým chybám při psaní kódu. Interoperabilita je další důležitou vlastností jazyka a umožňuje jeho použití v kódu napsaném v jazyce Java a naopak. Kotlin tak může využívat všechny existující funkce a knihovny jazyku Java [56].

Android Software Development Kit

Android Software Development Kit (SDK) je sada nástrojů a knihoven vyžadovaná pro vývoj mobilních aplikací. Díky Android SDK je proces vývoje plynulejší a jednodušší i pro začátečníky. Android SDK je optimalizován pro Android Studio, kde lze SDK nastavovat přes SDK Manager. Android SDK se skládá z několika částí, z nichž důležitá část je SDK Platform, která je číslovaná podle Android a API verze. SDK Platform musí být specifikován při vytváření aplikace čímž vývojář určí, jaký

typ sestavení (build) bude aplikace mít. Nejnovější verze SDK Platform poskytují více funkcí, avšak starší zařízení s nimi nemusí být kompatibilní.

Přehled důležitých verzí je zobrazen v tab.2.1. Android verze jsou pojmenovávány podle abecedního pořadí, kde jméno verze je inspirováno chutnými dobrotami [57, 58, 59].

Tab. 2.1: Přehled Android a API verzí.

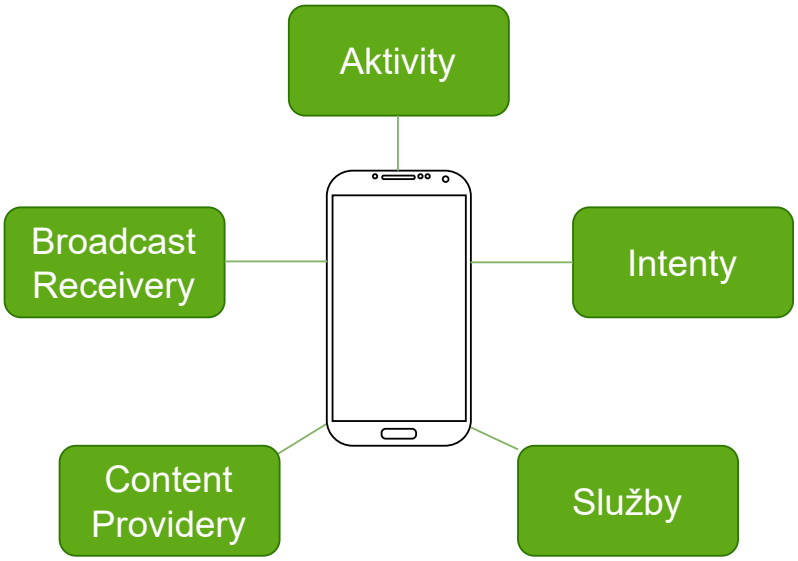
Android verze	Jméno	API verze	Měsíc a rok vydání
11	-	30	9/2020
10	-	29	9/2019
9	Pie	28	8/2018
8	Oreo	26	12/2017
7	Nougat	24	8/2016
6	Marshmallow	23	10/2015
5	Lollipop	21	10/2014
4.4	KitKat	19	10/2013
4.1	Jelly Bean	16	7/2012
4	Ice Cream Sandwich	14	10/2011
3	Honeycomb	11	2/2011
2.3	Gingerbread	9	12/2010
2.2	Froyo	8	5/2010
2	Eclair	5	10/2009
1.6	Donut	4	9/2009
1.5	Cupcake	3	4/2009
1.1	-	2	2/2009
1.0	-	1	9/2008

2.3 Základní bloky aplikace

Jako každá aplikace, tak i mobilní aplikace pro systém Android má své základní stavební bloky, které pomáhají učinit aplikaci funkční. Mezi základní bloky aplikace patří [60]:

- aktivity (Activities),
- Služby (Services),
- Broadcast Receivery (Broadcast Receivers),
- Content Providery (Content Providers),
- Intenty (Intents).

Jednoduché zachycení základních bloků mobilní aplikace pro systém Android je zobrazeno na obr. 2.3. Základní bloky musí být navíc vždy zmíněny ve speciálním souboru, který nese název `AndroidManifest.xml`.



Obr. 2.3: Základní bloky mobilní aplikace pro systém Android.

Aktivity

Aktivita slouží jako vstupní bod uživatelské interakce s mobilní aplikací. Aktivity definují, jakým způsobem bude uživatel aplikaci ovládat a jak bude aplikace reagovat na uživatelské vstupy. Obecně lze říci, že jedna aktivita implementuje jednu obrazovku aplikace. Aplikace se většinou skládá z více obrazovek, tedy i z více aktivit, což znamená, že jedna aktivita je vždy hlavní. Tato hlavní aktivita po spuštění aplikace zobrazuje první obrazovku, přes kterou se dá dále dostat na jiné obrazovky pomocí dalších aktivity, které jsou vnořeny v hlavní aktivitě. Všechny aktivity musí být zaznamenány v souboru AndroidManifest.xml [61].

Služby

Služba je komponenta, která provádí dlouhotrvající operace. Na rozdíl od aktivity, služby neposkytují uživatelské rozhraní. Po spuštění může služba běžet po definovaný čas, dokonce i po uživatelově přejití do jiné aplikace. Jakákoliv komponenta může být navázána na službu, aby s ní mohla komunikovat a provádět skrz ní meziprocesorovou komunikaci. V mobilní aplikaci mohou být implementovány tři typy služeb - služba v popředí (foreground), služba na pozadí (background) a vázaná služba (bound). Služba v popředí provádí viditelné operace a musí o svém běhu informovat uživatele

pomocí notifikace (oznámení). Služba na pozadí naopak provádí operace, které jsou uživateli skryté. Vázaná služba běží tak dlouho, dokud běží s ní svázaná komponenta [62].

Broadcast Receiver

Komponenta nazývaná broadcast receiver reaguje na systémová oznámení nebo oznámení jiných aplikací jako jsou například slabá baterie nebo zapojení sluchátek do zařízení. Na základě reakce této komponenty na danou operaci může být vyvolána akce v aplikaci [63].

Content Provider

Content Provider by se dal přeložit jako poskytovatel obsahu. Tato komponenta tedy poskytuje obsah jiným aplikacím, přičemž data jsou zapouzdřena a dále předána aplikacím pomocí jednoduchého rozhraní. Content provider je potřebný ve chvíli, kdy si aplikace mezi sebou chtějí poskytnout obsah, což může nastat například ve chvíli, kdy aplikace chce získat seznam kontaktů z aplikace Kontakty [60, 64].

Intent

Intent je jednoduchý objekt, který reprezentuje záměr (intention) provést nějakou akci. Pomocí intentu mohou být spouštěny aktivity, může být prováděna komunikace se službami nebo může být využit u broadcast receiverů. Intent je například použit v aplikaci, která chce zobrazit webovou stránku. Aplikace vyvolá záměr zobrazení stránky a tím vytvoří intent, který je předán systému. Systém vyhledá v části kódu, která aplikace umí intent obsloužit, a na základě toho spustí webový prohlížeč [60, 65].

AndroidManifest.xml

Soubor AndroidManifest.xml je nezbytnou součástí každé mobilní aplikace a musí být umístěn v kořenovém adresáři projektu. V tomto souboru se nachází podstatné informace o mobilní aplikaci, které jsou dále předávány sestavovacím nástrojům, operačnímu systému a obchodu Google Play.

První hodnota, která musí být v souboru deklarována je název balíčku. Název balíčku je využíván sestavovacími nástroji za účelem určení umístění částí kódu. Při vytváření balíčku aplikace je tato hodnota nahrazena identifikačním kódem (ID) aplikace, který je dále použit jako jedinečný identifikátor v operačním systému a v obchodu Google Play.

Součástí souboru musí také být všechny použité komponenty, mezi které se řadí aktivity, služby, broadcast receivery a content providery. U každé komponenty pak musí být definované její základní vlastnosti.

Dále jsou v souboru sepsána všechna povolení umožňující aplikaci přístup do zabezpečených částí operačního systému nebo k jiným aplikacím. Součástí mohou být i povolení, která musí mít jiné aplikace, aby mohly přistupovat k vytvářené aplikaci.

Posledními důležitými hodnotami v souboru jsou informace o vlastnostech hardwaru a softwaru, které jsou aplikací vyžadovány pro její správné spuštění [66].

3 Vlastní návrh aplikace

V rámci návrhu musela být provedena analýza zranitelností, které má vyvíjená aplikace prezentovat. V této části jsou tedy zanalyzovány zranitelnosti exportované aktivity, chybějící autorizace, broadcast receiverů, služeb, interního úložiště a neoprávněného přístupu k datům včetně zranitelností Path Traversal a SQL Injection.

3.1 Analýza zranitelností

Zranitelnost exportované aktivity

Aktivita může být definována mnoha parametry. Jedním z nich je parametr `exported`, který nabývá hodnot `true` nebo `false`. Tento parametr udává, zda daná aktivita může by spuštěna jinou aplikací. Pokud je parametr nastaven na `false`, aktivita může být spuštěna pouze komponentami stejné aplikace, ve které se aktivita nachází. V případě nastavení parametru na hodnotu `true`, je povoleno aktivitu spustit jakoukoli jinou aplikací. Parametr by měl být nastaven na hodnotu `true` ve chvíli, kdy jsou v aplikaci využívány intenty.

Aplikace využívající exportovanou aktivitu je zranitelná tím, že ke všem jejím komponentám může být přistoupeno jakoukoliv aplikací. Aktivita může být spuštěna i škodlivou aplikací, pro kterou je díky tomuto nastavení snadné získat citlivá data nebo modifikovat vnitřní stav aplikace. Do Android verze 4.2 jsou všechny komponenty včetně aktivit ve výchozím stavu exportované [67, 68].

Chybějící autorizace

Chybějící nebo nezabezpečená autorizace již byla rozebrána v předchozí kapitole, avšak zde je rozebrána v souvislosti s vyvíjenou mobilní aplikací. Chybějící autorizace je nejvíce zmiňována v souvislosti s API voláním a s také již zmíněnou zranitelností IDOR. Zranitelnost IDOR využívá faktu, že je k objektům přistupováno přímo, nejčastěji pomocí ID uživatele. Stejné ID je použito jako primární klíč databáze k přístupu k tabulce obsahující uživatelské informace. Pokud se v URL pro získání konkrétních uživatelských dat objeví např. „ID=123“, může být toto ID v dalším kroku změno na „ID=124“ a uživatel tak dostane informace úplně jiného uživatele. To samé platí pro API požadavek typu GET například ve tvaru `/api/message/users/userid`, kde za `userid` může být dosazeno jakékoliv číslo pro získání uživatelských dat jakéhokoliv uživatele.

S chybějící autorizací, ale především s autentizací, souvisí tzv. enumerace účtů. Jedná se o aktivitu, při které běžný uživatel může zjistit, jací další uživatelé jsou

registrováni v aplikaci (na serveru). Uživateli stačí vyplnit přihlašovací pole nějakým jménem a zaslat tento požadavek na server. V případě, že uživatel existuje, je vrácena jiná odpověď než v případě, že uživatel neexistuje. Kdokoliv tak může získat seznam uživatelů a dále tyto informace zneužít ve svůj prospěch. K seznamu uživatelů by se měl dostat pouze administrátor [69, 70].

Neautorizovaný přístup je dále využit ve zranitelnosti LFD (Local File Disclosure). Tato zranitelnost je také spojena s API voláním a umožňuje útočníkům stahovat data z webové aplikace díky zneužití funkcí, které slouží pro stahování souborů. Útočník se tak může dostat k souborům, ke kterým má přístup pouze administrátor. LFD zranitelnost plyne z nesprávné implementace kódu pro běh serveru [71].

U špatně implementovaného API volání pro stahování souborů může být zneužita zranitelnost RFD (Reflected File Download). Útočník je schopen pomocí této zranitelnosti získat neautorizovaný přístup do zařízení oběti [72].

V neposlední řadě může být neautorizovaný přístup využit pro vzdálené spuštění kódu na serveru. Tato zranitelnost je známá pod zkratkou RCE (Remote Code Execution). Zranitelnost plyne například z nedostatečného ověření vstupů nebo chybné konfigurace serveru [73].

Zranitelnost broadcast receiverů a služeb

Broadcast receivery a služby se řadí společně s aktivitami mezi základní komponenty aplikace. Stejně jako u aktivit je možné i u těchto komponent nastavovat různé parametry včetně parametru `exported`. Obě komponenty se chovají při nastavení tohoto parametru stejně jako aktivity. V případě, že musí být komponenty exportované, například kvůli využití intentů, je nutné nastavit jejich oprávnění.

Všechna oprávnění se nastavují buď na úrovni komponenty nebo na úrovni aplikace. U komponent je oprávnění nastavováno pomocí parametru `permission`. Správné nastavení tohoto parametru bude vést k tomu, že přístup k dané komponentě budou mít pouze aplikace s oprávněním a žádné jiné. Nemůže tak dojít k tomu, že jiná aplikace bude moct spustit funkci komponenty. Pokud komponenta nemá explicitně nastavené oprávnění, dědí oprávnění definované v manifestu aplikace. Aplikace si pak u uživatele žádá různá oprávnění, například zda může zasílat zprávy [74, 75, 76].

Zranitelnost interního úložiště

Operační systém android poskytuje dva typy úložišť - externí a interní. Mobilní aplikace jsou ve výchozím nastavení ukládány na interní úložiště zařízení, kde se také nacházejí všechna uložená data aplikace. Přístup k těmto datům by tak měla mít pouze daná aplikace a žádná jiná. Nicméně, může nastat situace, ve které je

potřeba poskytnout data jiné aplikaci. Pro zachování bezpečnosti aplikace by se tento problém měl řešit pomocí content providerů, ale existuje ještě možnost nastavení oprávnění přímo žádaným souborům. Žádanému souboru je v kódu aplikace přidělen parametr `MODE_WORLD_WRITABLE` pro případ jeho úpravy nebo parametr `MODE_WORLD_READABLE` pro možnost jeho přečtení. Přístup k souborům na interním úložišti implementující tyto parametry není pro útočníky příliš obtížný, protože tento typ implementace neposkytuje možnost omezení přístupu k datům konkrétním aplikacím. Jakákoliv aplikace tak může podle nastavených oprávnění manipulovat s obsahem nebo jej číst [74, 77, 78].

Neoprávněný přístup k datům

Přístup k datům se zpravidla řeší pomocí content providerů, které poskytují kontrolu nad oprávněními pro řízení přístupu. Špatná implementace content providerů v kódu aplikace může způsobit neoprávněný přístup k datům jak v rámci dané aplikace, tak pomocí jiné aplikace.

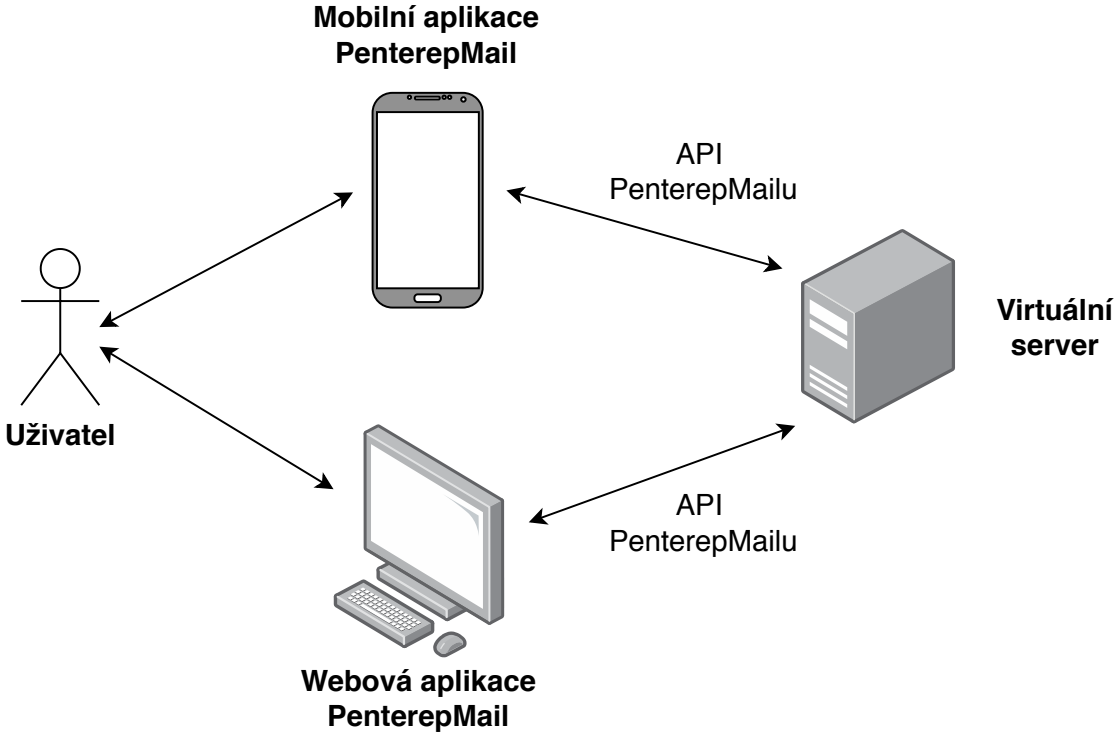
U content providerů může být využito zranitelnosti SQL Injection, která spočívá ve vložení specifického SQL dotazu přes neošetřený vstup. Úspěšný útok povede například ke čtení citlivých dat z databáze nebo k modifikaci databáze. Zranitelnost může být využita pokud se požadavky na dané akce s databází nezasílají pomocí primárních metod content providerů, jako jsou metody `query()` nebo `update()`. K využití zranitelnosti může dojít i přes použití primárních metod v případě, že argument `projection` pro výběr sloupců nebo argument `selection` pro výběr řádků bude vytvořen zřetězením uživatelských dat před jejich odesláním metodě [79].

Pro úspěšný přístup ke konkrétnímu souboru pomocí zneužití SQL Injection může být potřeba znát celou cestu k tomuto souboru. Útočník pro získání znalosti o celé cestě může zneužít zranitelnost FPD (Full Path Disclosure), která útočníkovi zobrazí celou cestu k souboru při nevhodné implementaci API volání. Celá cesta může být například ukázána v chybové zprávě v odpovědi, která byla vrácena po volání konkrétního API požadavku [80].

Další zranitelností content providerů je zranitelnost Path Traversal. Tento útok je založen na zisku neoprávněného přístupu k souborům a adresářům, které jsou uloženy mimo `root` adresář. Specifikováním absolutní cesty k souboru je možné dostat se ke kódům aplikace, konfiguračním souborům nebo výpisu uživatelů a jejich hesel. Aplikace je zranitelná na tento útok ve chvíli, kdy jsou v kódu aplikace špatně implementovány metody content providerů pro otevření souborů, jako jsou metody `openFile()` a `openAssetFile()`. Tyto metody musí správně ověřovat příchozí URI (Uniform Resource Identifier) parametry pro snížení rizika využití této zranitelnosti [74, 81, 82, 83].

3.2 Návrh základních bloků aplikace

Mobilní aplikace byla vytvářena jako součást projektu PenterepMail, který je stále ve fázi vývoje a který bude sloužit jako doplněk vyvíjeného projektu zaměřeného na penetrační testování Penterep. Celý projekt PenterepMail bude po dokončení obsahovat zranitelnou webovou a mobilní aplikaci ve formě mailového klienta. Obě aplikace budou komunikovat s virtuálním serverem pomocí API PenterepMailu. Obr. 3.1 zachycuje blokový návrh komunikace všech částí projektu PenterepMail. Webová aplikace i server jsou navrženy tak, aby trpěly velkým množstvím zranitelností [84]. Cílem této části bylo navrhnout mobilní aplikaci, která by prezentovala jak zranitelnosti mobilní aplikace tak zranitelnosti při komunikaci se serverem.



Obr. 3.1: Blokový návrh komunikace.

Mobilní aplikace byla navržena tak, aby fungovala jako mailový klient. Uživatel bude mít možnost aktivování aplikace, přihlášení se a odhlášení se, prohlížení přijatých a doručených zpráv, zasílání nových zpráv a spravování kontaktů včetně jejich přidávání a odebírání. Všechny tyto funkcionality budou doprovázeny záměrnou implementací zranitelností, které vycházejí z výše uvedené analýzy.

Během první fáze návrhu byly vytvořeny obrazovky před přihlášením a po přihlášení, které jsou zobrazeny na obr. 3.2 a obr. 3.3. Po implementaci těchto obrazovek bylo zjištěno, že by se uživatel musel stále překlikávat do menu, pokud by chtěl

vyvolat jinou aktivitu. Z toho důvodu byl návrh pozměněn a součástí všech aktivit dostupných po přihlášení se stalo tzv. zajížděcí menu, které uživateli umožní pohodlnější ovládání aplikace.

Obr. 3.2: Původní návrh obrazovky před přihlášením.

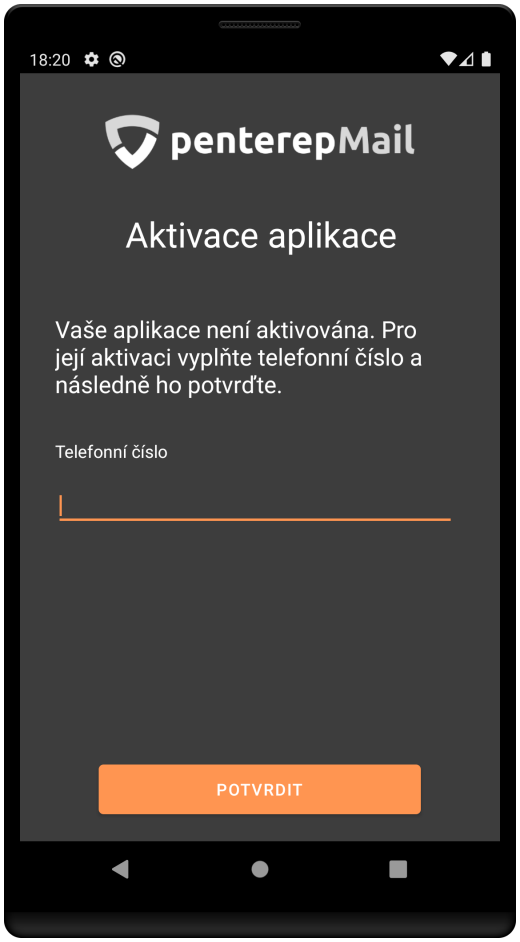
Obr. 3.3: Původní návrh obrazovky po přihlášení.

V následujícím textu jsou zobrazeny jednotlivé obrazovky finální aplikace a dále je zde pouze stručně popsán jejich účel a činnost. Konkrétní popis implementace jednotlivých komponent je součástí kapitoly Vlastní implementace mobilní aplikace.

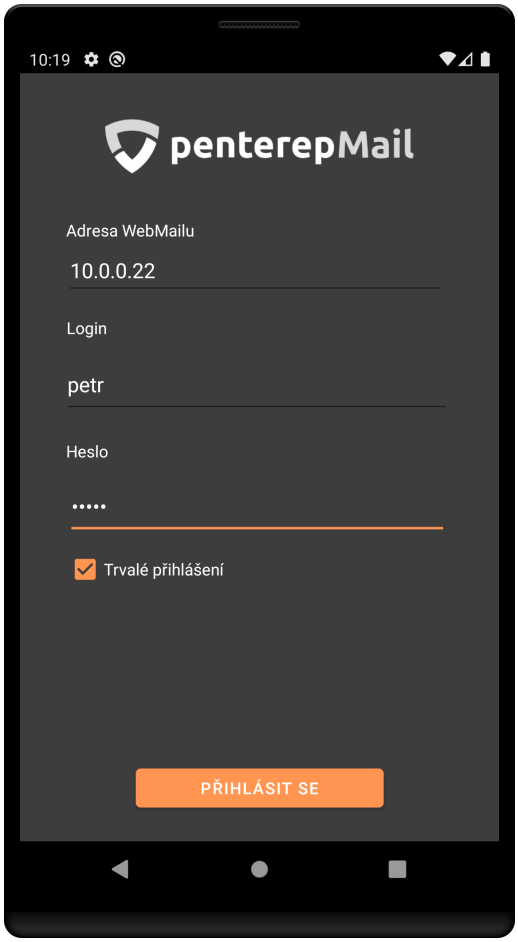
Po spuštění mobilní aplikace se objeví aktivační obrazovka (obr. 3.4), přes kterou musí uživatel zadat své telefonní číslo. Aplikace pošle na zadané číslo SMS s potvrzovacím kódem, který byl vygenerován při události pro zaslání SMS a který byl uložen do souboru aplikace. Pokud se uložený kód v aplikaci a kód zaslaný v SMS shodují, uživatel je vpuštěn do aplikace. Pokud aktivace proběhla jednou úspěšně, tato obrazovka již při dalším spuštění není zobrazena.

První obrazovkou po aktivaci je přihlašovací obrazovka (obr. 3.5). Skrze tuto obrazovku je nutno pro úspěšné přihlášení zadat IP adresu WebMailu, neboli virtuálního serveru, na kterém běží obraz PenterepMail. Dále uživatel zadává svůj login

a své heslo. V případě, že chce být trvale přihlášen, zatrhne zaškrtávací pole s názvem „Trvalé přihlášení“. Po stisknutí tlačítka „Přihlásit se“ je uživatel úspěšně či neúspěšně přihlášen do aplikace.



Obr. 3.4: Aktivační obrazovka

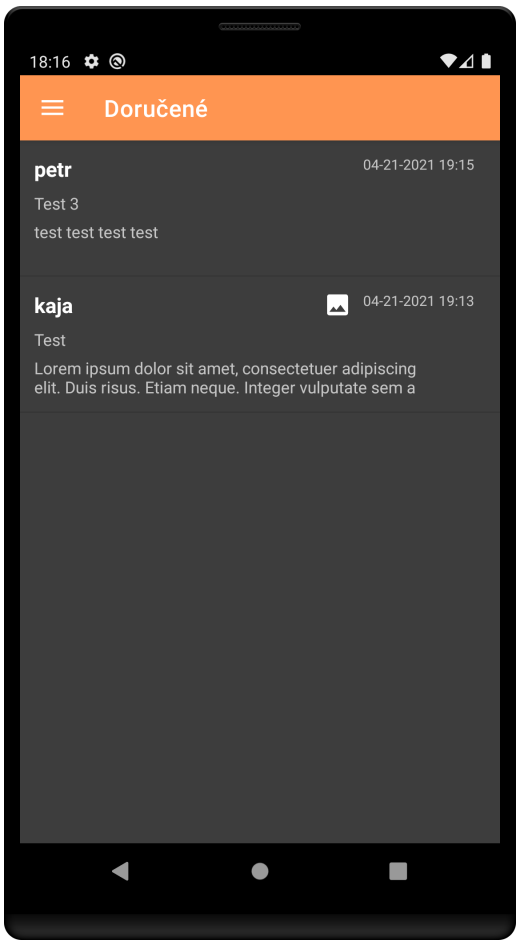


Obr. 3.5: Přihlašovací obrazovka

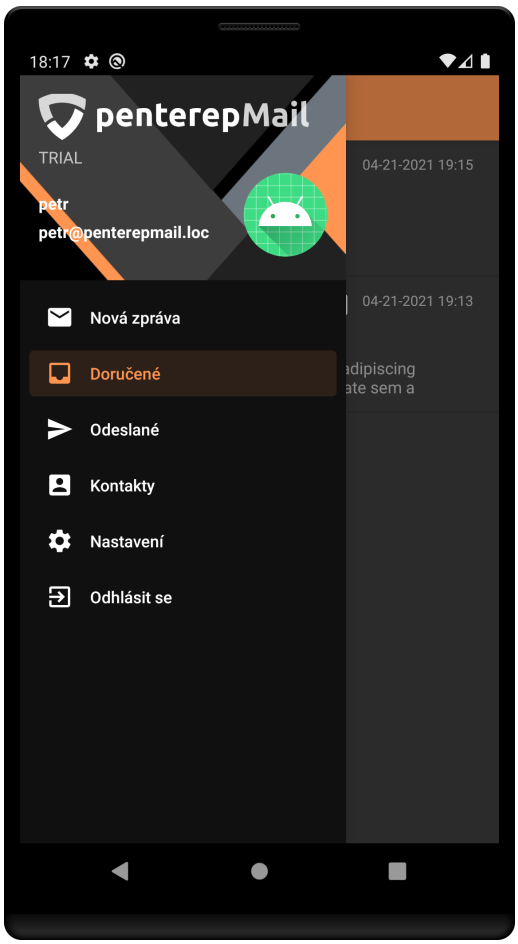
V případě úspěšného přihlášení je uživatel vždy přesměrován na obrazovku s doručenou poštou (obr. 3.6). Uživateli se na této obrazovce zobrazují doručené emaily, kde je u každého z nich zobrazen odesílatel, předmět, první dva řádky textu zprávy a datum doručení. V případě, že je součástí emailu také příloha, je tento fakt zobrazen pomocí ikony obrázku. Stejně vypadající je také obrazovka pro odeslanou poštu s tím rozdílem, že u každého emailu je místo odesílatele zobrazen email příjemce.

Po kliknutí na položku s konkrétní zprávou se zobrazí její detail (obr. 3.8). V detailu zprávy se uživateli ukazuje odesílatelův login a email, předmět zprávy, datum odeslání zprávy a celý text zprávy. Podle počtu příloh se všechny přílohy zobrazí v dolní části obrazovky. Uživatel má také možnost skrze tuto obrazovku zprávu smazat nebo si otevřít obsah konkrétní přílohy.

Pro rychlou změnu obrazovek je uživateli k dispozici zajížděcí menu (obr. 3.7). Uživatel má možnost skrze menu dostat se na obrazovku pro odeslání nové zprávy a zobrazit si doručenou a odeslanou poštu, kontakty a nastavení. Uživatel se dále může odhlásit z aplikace. Toto menu může být zobrazeno na jakékoliv obrazovce v rámci úspěšného přihlášení. Menu se zobrazuje kliknutím na ikonu tří vodorovných čar v levé horní části aplikace nebo tažením prstu po obrazovce z levé části do pravé. Naopak, zavírání probíhá kliknutím mimo menu nebo tažením prstu po obrazovce z pravé do levé části. V hlavičce menu je zobrazen uživatelův login, email a profilový obrázek. Pokud si uživatel žádný profilový obrázek nenastavil, je zobrazen výchozí obrázek s logem systému Android. Součástí hlavičky je také logo a verze, kterou uživatel používá. Ve výchozím nastavení je verze vždy „trial“.



Obr. 3.6: Obrazovka s doručenou poštou



Obr. 3.7: Obrazovka s hlavní nabídkou

Obrazovka „Kontakty“ poskytuje přehled všech uložených kontaktů uživatele (obr. 3.9). U každého kontaktu se zobrazuje jméno, email, popis a ikona koše pro jeho smazání. V rámci kontaktů je také umožněno jejich vyhledávání pomocí vy-

hledávacího okna v horní části obrazovky. Uživatel má dále možnost kontakt přidat pomocí tlačítka „Přidat kontakt“. Tato možnost je ale k dispozici pouze v plné (full) verzi. Obrazovka pro přidání kontaktu obsahuje stejná pole pro vyplnění, jako jsou zobrazena v přehledu kontaktů.

Pro odeslání nové zprávy slouží obrazovka „Nová zpráva“. Uživatel musí zadat příjemce a předmět. Text zprávy je pak volitelnou položkou. Zpráva je dle vyplněných údajů odeslána po stisknutí tlačítka „Odeslat zprávu“. V případě, že není vyplněn email nebo předmět, zobrazí se hláška informující uživatele o neodeslání zprávy. Podobná hláška je ukázána při vyplnění neplatného emailu.

V rámci nastavení je zobrazen přehled momentálních hodnot, se kterými aplikace pracuje. Uživatel se zde dozví IP adresu WebMailu, verzi aplikace, login a email. Verze aplikace může být změněna pouze v kódu aplikace pomocí využití jedné ze zranitelností, která bude popsána dále v textu.

Poslední položkou menu je položka pro odhlášení se. Po kliknutí na „Odhlásit se“ je uživatel odhlášen a přesměrován na přihlašovací obrazovku. Také jsou vymazány všechny přihlašovací a další údaje z aplikace.

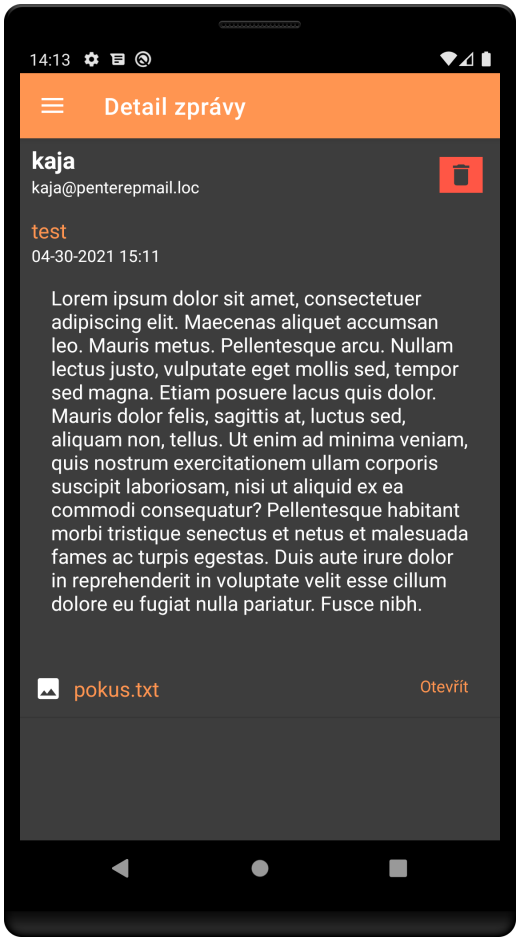
Součástí mobilní aplikace není registrace nového uživatele, změna profilového obrázku nebo odeslání přílohy u nové zprávy. Všechny tyto akce musí uživatel provést pomocí webové aplikace.

3.3 Výběr vhodných nástrojů a technologií

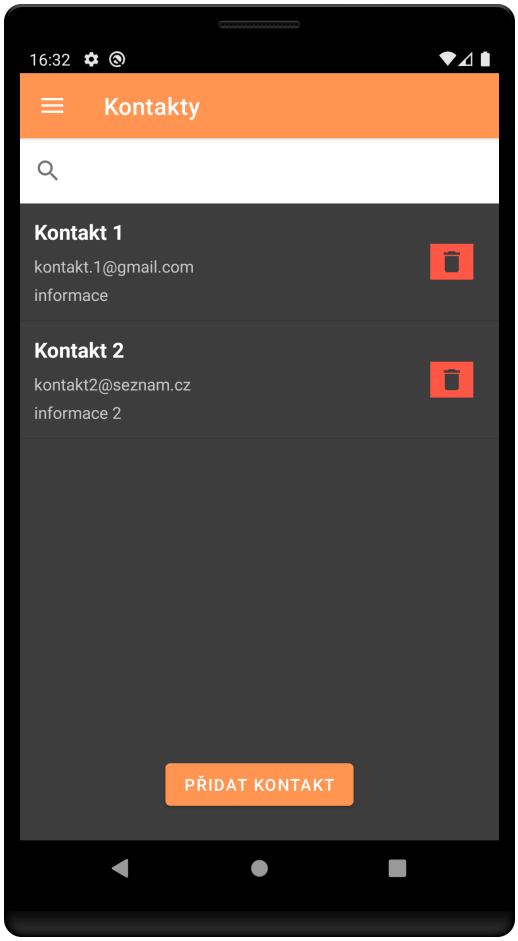
V rámci fáze analýzy a návrhu byly definovány nástroje a technologie, které byly použity pro implementaci aplikace. Virtuální obraz serveru ve verzi 1.2 byl distribuován ve formátu `.ova` (Open Virtualization Format), který se importoval pomocí virtualizačního nástroje Oracle VM VirtualBox ve verzi 6.1. Virtualizační nástroj běžel na osobním notebooku značky Dell s operačním systémem Windows 10 Home, procesorem Intel Core i7 a pamětí RAM (Random Access Memory) 16 GB.

Aplikace byla vyvíjena v Android verzi 4.1 (API 16) z důvodu jasnější demonstrace zranitelností. U vyšších API jsou některé typy implementací vedoucí ke zranitelnosti již potlačeny a při sestavování aplikace zobrazí výjimku. Díky zvolené nižší verzi API mohou být zranitelnosti demonstrovatelné i na novějších verzích operačního systému Android.

Pro vývoj aplikace bylo zvoleno vývojové prostředí Android Studio ve verzi 4.1.1. Pro implementaci byl použit programovací jazyk Java ve verzi 1.8.0 a pro ukládání strukturovaných dat v případě kontaktů či zpráv byla využita databáze SQLite. V rámci Android Studia byl rozběhnut emulátor Pixel 2 s SDK platformou ve verzi 10 (API 29). Vývoj aplikace probíhal na výše zmíněném osobním notebooku. Výsledná aplikace je sestavena do spustitelného souboru typu `.apk`.



Obr. 3.8: Obrazovka s detailem zprávy



Obr. 3.9: Obrazovka s kontakty

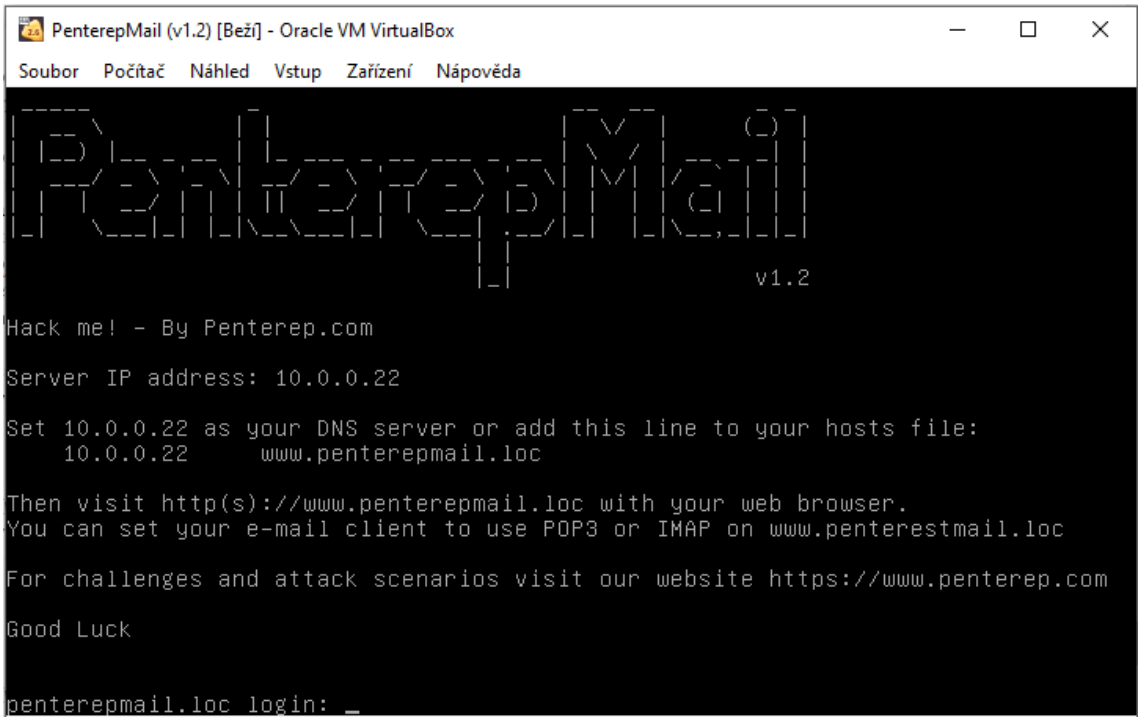
4 Vlastní implementace mobilní aplikace

V této kapitole je popsána příprava prostředí pro implementaci a poté implementace samotná. Popis implementace je vždy rozdělen podle aktivit, u kterých je detailně popsán jejich účel a činnost.

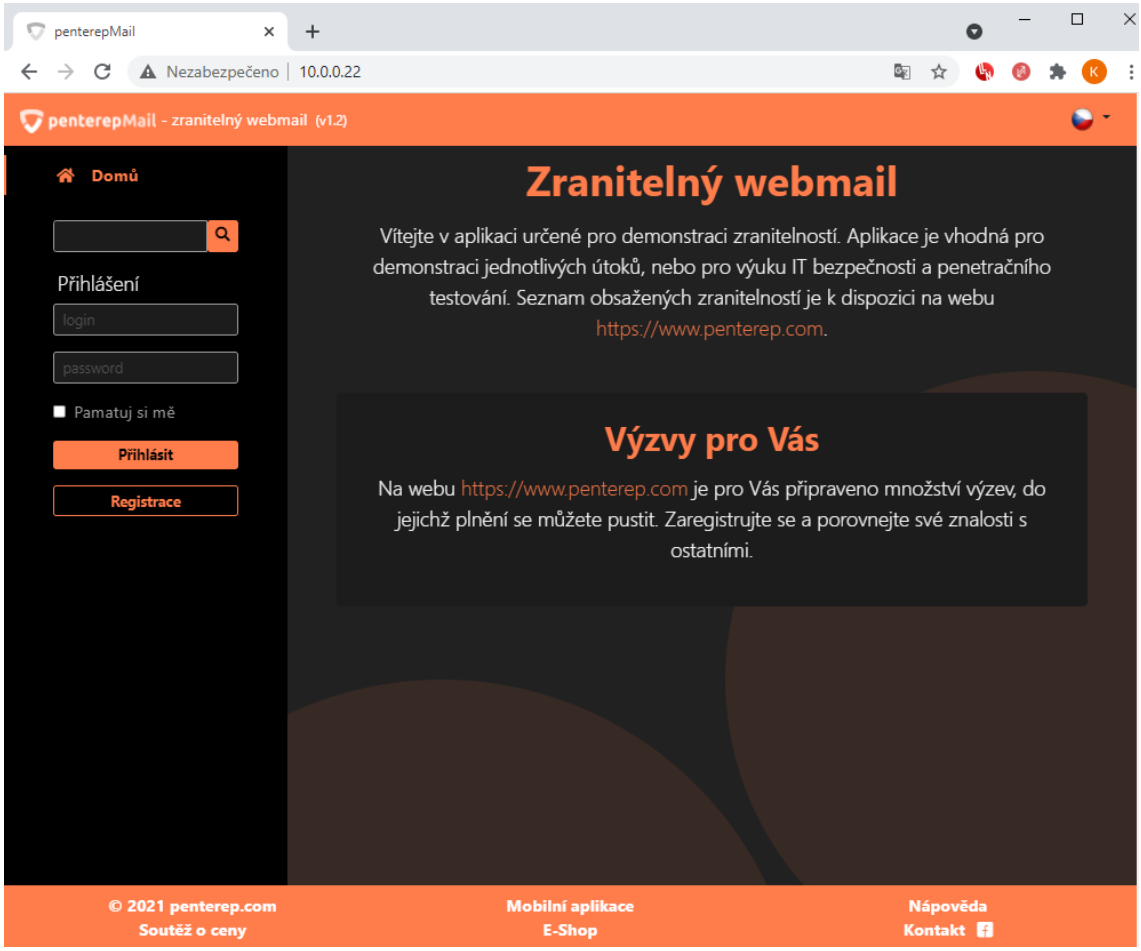
4.1 Příprava prostředí

Prvním krokem při přípravě prostředí bylo stažení obrazu virtuálního serveru ze stránky `https://www.penterep.com`. Tento obraz byl nejdříve naimportován pomocí virtualizačního nástroje a dále byla virtuálnímu stroji nastavena síťová karta na možnost síťový most. Po spuštění a načtení stroje se zobrazila jemu přiřazená IP adresa na základě protokolu DHCP (Dynamic Host Configuration Protocol). Po zadání této IP adresy do webového prohlížeče byla načtena webová aplikace. Běžící virtuální stroj je zobrazen na obr. 4.1 a webová aplikace na obr. 4.2. Tímto bylo ověřeno, že komunikace mezi osobním notebookem a serverem je funkční.

Další krok přípravy spočíval v rozběhnutí vývojového prostředí Android Studio včetně emulátoru. Spuštěné vývojové prostředí s emulátorem je zachyceno na obr. 4.3.



Obr. 4.1: Běžící virtuální stroj s přidělenou IP adresou.



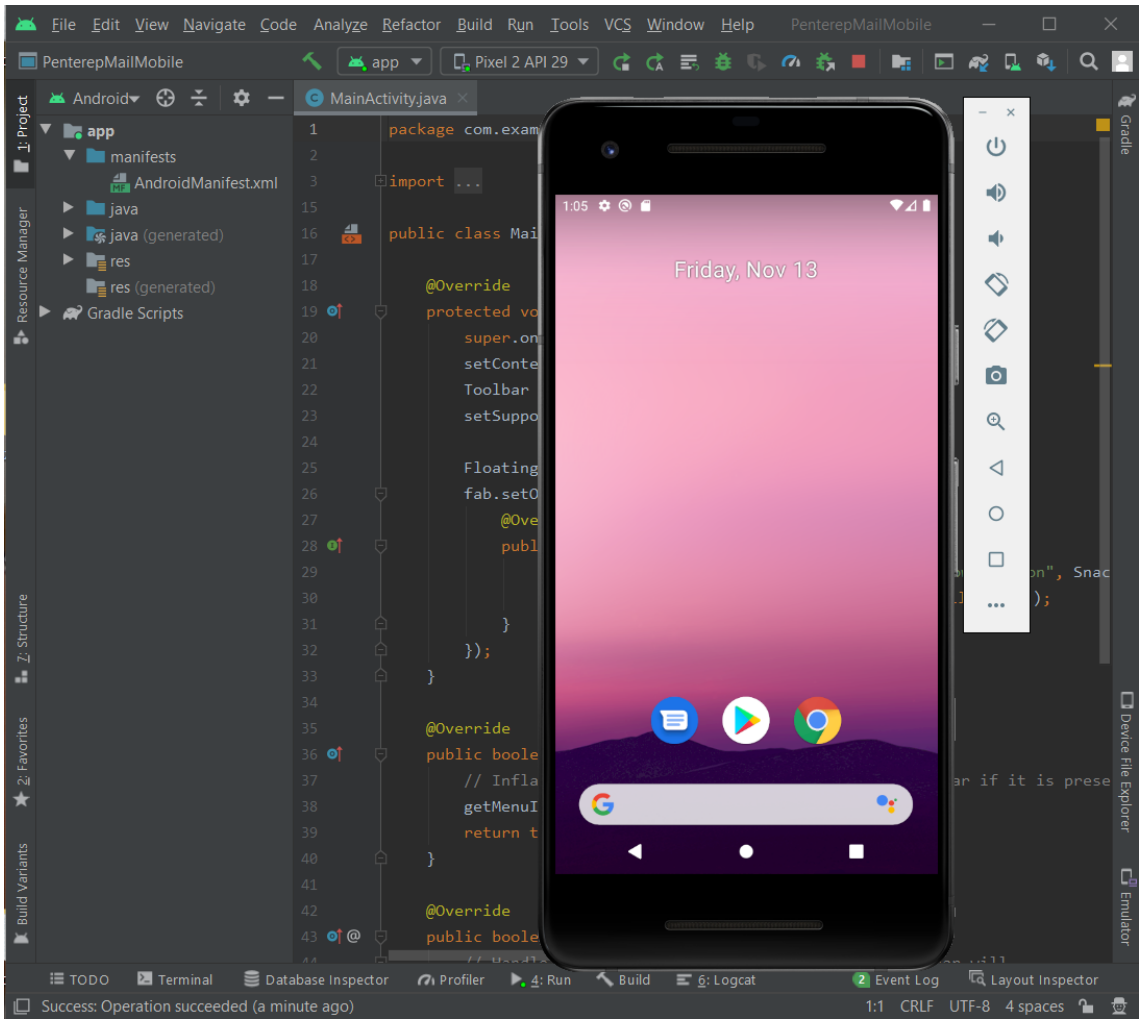
Obr. 4.2: Běžící webová aplikace na dané IP adrese.

4.2 Struktura aplikace

Pro lepší pochopení následného popisu implementace je zapotřebí zmínit strukturu aplikace a její důležité soubory, které byly během implementace vytvořeny a využity.

Aplikace byla vytvářena jako projekt s názvem `PenterepMailMobileApp`, jehož součástí je balíček `com.example.penterepmailmobileapp`, ve kterém se nachází všechny aktivity a třídy aplikace. Aplikace je rozdělena na osm aktivit, z nichž každá nese název jejího účelu v angličtině:

- `ActivationActivity.java` - aktivační aktivita.
- `ContactsActivity.java` - aktivita pro zobrazení kontaktů.
- `InboxSentActivity.java` - aktivita pro zobrazení doručených a odeslaných zpráv.
- `LoginActivity.java` - přihlašovací aktivita.
- `MessageDetailActivity.java` - aktivita pro zobrazení detailu zprávy.
- `NewContactActivty.java` - aktivita pro přidání nového kontaktu.



Obr. 4.3: Vývojové prostředí Android Studio s emulátorem.

- `NewMessageActivty.java` - aktivita pro zaslání nové zprávy.
- `SettingsActivty.java` - aktivita pro zobrazení nastavení.

V aplikaci jsou také implementované další třídy, které jsou zmíněny dále v textu buď v rámci popisu aktivit nebo samostatně.

Nedílnou součástí projektu je složka `res` s cestou `\app\src\main\res`, ve které se nachází zdrojové soubory, většinou ve formátu `.xml`, využité v aplikaci. Např. ve složce `\drawable` se nachází ikony, logo a pozadí, ve složce `\layout` najdeme rozvržení a definice vzhledu jednotlivých aktivit a ve složce `\values` jsou zaznamenány hodnoty barev a textových řetězců. Složka `\mipmap` obsahuje různé velikosti ikony aplikace, která se zobrazuje v hlavní nabídce telefonu. Výhodou používání zdrojů je fakt, že například konkrétní barvu definujeme pouze jednou v soubor `colors.xml` a v aplikaci se už odkazujeme pouze na toto místo. Odpadá tak práce stálého vepisování hodnoty barvy.

Velmi důležitou součástí je již zmiňovaný `AndroidManifest.xml`, jehož části obsahu jsou popsány také dále v textu. Ve struktuře aplikace najdeme další složky a soubory, avšak k popisu implementace nejsou stěžejní. Nicméně jsou stále nedílnou součástí aplikace a jejich nepřítomnost v projektu by vedla k chybě při spuštění.

Aplikace je k dispozici v českém jazyce, avšak texty, které jsou implementovány přímo na straně serveru jsou v anglickém jazyce (například texty ve vyskakovacích oknech).

4.3 Použitá API volání

Pro komunikaci aplikace s virtuálním serverem jsou použita API volání implementována na straně serveru. Návrh API volání nebyl součástí této práce, avšak jejich použití v mobilní aplikaci slouží k demonstraci jejich záměrně chybné implementace, a tedy i k ukázce jejich zranitelností.

API volání, účel volání, typ požadavku, vrácená odpověď (zmíněny pouze názvy vrácených hodnot) a zranitelnost jsou pro jednotlivá volání sepsána v tabulkách tab. 4.1, tab. 4.2, tab. 4.3, tab. 4.4, tab. 4.5, tab. 4.6, tab. 4.7, tab. 4.8, tab. 4.9, tab. 4.10 a tab. 4.11.

Odpovědi jsou vráceny v typu JSON (JavaScript Object Notation) a jednotlivé názvy hodnot odpovědi jsou zapsány v angličtině. Avšak pro lepší srozumitelnost jsou v následujících tabulkách názvy hodnot uvedeny v češtině. Ve vrácené odpovědi se vždy vrací „status“, který udává, zda byl nebo nebyl požadavek v pořádku. Hodnota „zpráva“ zase nese zprávu, která popisuje událost vyvolanou zpracováním zaslaného požadavku.

Veškerá komunikace s API využívá protokolu HTTP a je tedy nešifrovaná. Před každým api voláním je v aplikaci nutno specifikovat celou URL cestu, která vypadá takto: `http://[IP adresa WebMailu]/api/[konkrétní volání]`. Zasílání požadavků na danou URL je v aplikaci implementováno pomocí knihovny `volley`, která umožňuje rychlou a jednoduchou komunikaci po síti [85].

V hlavičce všech API požadavků je posíláno při přihlášení vygenerované cookie ID, nebo také session ID, které slouží k udržení spojení přihlášeného uživatele se serverem. V obsahu hlavičky požadavků se dále posílá typ zasílaného obsahu, tzv. „content type“, který má hodnotu `application/x-www-form-urlencoded`.

4.4 Implementace jednotlivých aktivit a tříd

Na úvod této části je popsán způsob ukládání dat, které jsou potřebné pro chod aplikace. Dále v textu jsou již uvedeny samotné aktivity. Na konci této části jsou

Tab. 4.1: API pro přihlášení do aplikace

API volání
api/login
Typ požadavku
POST
Zasílané parametry
login=admin&heslo=Admin123
Účel
Přihlášení se do aplikace.
Vrácená odpověď - OK
status, zpráva, login, email, ID uživatele, profilový obrázek, session ID
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
SQL Injection, enumerace účtů, hádání hesla (není omezen počet pokusů)

Tab. 4.2: API pro odhlášení se z aplikace

API volání
api/logout
Typ požadavku
GET
Účel
Odhlášení se z aplikace.
Vrácená odpověď - OK
status, zpráva
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
bez zranitelností

Tab. 4.3: API pro načtení doručených zpráv konkrétního uživatele

API volání
api/messages/INBOX/[ID uživatele]
Typ požadavku
GET
Účel
Načtení doručených zpráv uživatele s daným ID uživatele.
Vrácená odpověď - OK
Seznam doručených zpráv, kde každá z nich obsahuje: ID zprávy, datum, odesílatele, email odesílatele, příjemce, email příjemce, předmět, obsah zprávy, seznam příloh
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
SQL Injection, IDOR

Tab. 4.4: API pro načtení odeslaných zpráv konkrétního uživatele

API volání
api/messages/Sent/[ID uživatele]
Typ požadavku
GET
Účel
Načtení doručených zpráv uživatele s daným ID uživatele.
Vrácená odpověď - OK
Seznam doručených zpráv, kde každá z nich obsahuje: ID zprávy, datum, odesílatele, email odesílatele, příjemce, email příjemce, předmět, obsah zprávy, seznam příloh
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
SQL Injection, IDOR

Tab. 4.5: API pro načtení obsahu přílohy

API volání
api/messages/[složka]/[ID uživatele]/[ID zprávy]/[jméno souboru]
Typ požadavku
GET
Účel
Načtení obsahu přílohy pro konkrétní zprávu (ID zprávy) konkrétního uživatele (ID uživatele) v dané složce (INBOX nebo Sent).
Vrácená odpověď - OK
Obsah přiloženého souboru.
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
FPD v chybové zprávě, RCE v /tmp adresáři

Tab. 4.6: API pro odeslání nové zprávy

API volání
api/newmessage/[ID uživatele]
Typ požadavku
POST
Zasílané parametry
předmět=test&příjemce=admin@penterepmail.loc&zpráva=text
Účel
Odeslání nové zprávy uživatelem s ID uživatele.
Vrácená odpověď - OK
status, zpráva
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
bez zranitelnosti

Tab. 4.7: API pro přidání nového kontaktu

API volání
api/newcontact/[ID uživatele]
Typ požadavku
POST
Zasílané parametry
email=test@foo.loc&jméno=FOO&popis=text
Účel
Přidání nového kontaktu uživatelem s ID uživatele.
Vrácená odpověď - OK
status, zpráva
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
bez zranitelnosti

Tab. 4.8: API pro načtení kontaktů

API volání
api/contacts/[ID uživatele]
Typ požadavku
GET
Účel
Načtení všech zpráv konkrétního uživatele (ID uživatele).
Vrácená odpověď - OK
Seznam kontaktů, kde každý z nich obsahuje: ID kontaktu, jméno, email, popis.
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
SQL Injection, IDOR (čtení cizích kontaktů změnou ID)

Tab. 4.9: API pro načtení profilového obrázku

API volání
api/messages/portrait/[ID uživatele]/[jméno souboru]
Typ požadavku
GET
Účel
Načtení profilového obrázku konkrétního uživatele (ID uživatele).
Vrácená odpověď - OK
Portrét ve formě obrázku
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
LFD, RFD skrze jméno souboru, Path Traversal

Tab. 4.10: API pro odstranění zprávy

API volání
api/messages/[složka]/[ID uživatele]/[ID zprávy]
Typ požadavku
DELETE
Účel
Odstranění konkrétní zprávy (ID zprávy) konkrétního uživatele (ID uživatele) v dané složce (INBOX nebo Sent).
Vrácená odpověď - OK
status, zpráva
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
bez zranitelnosti

Tab. 4.11: API pro odstranění kontaktu

API volání
api/contacts/[ID uživatele]/[ID kontaktu]
Typ požadavku
DELETE
Účel
Odstranění konkrétního kontaktu (ID kontaktu) konkrétního uživatele (ID uživatele).
Vrácená odpověď - OK
status, zpráva
Vrácená odpověď - Chyba (Error)
status, zpráva
Zranitelnost
bez zranitelnosti

uvedeny ostatní třídy implementované v aplikaci. Záměrně implementované zranitelnosti jsou vždy popsány u aktivit nebo tříd, ke kterým se vážou. Otestování a ukázka některých zranitelností je pak součástí poslední kapitoly s názvem Testování aplikace. První zranitelnost aplikace spočívá v povolení debuggingu při běhu aplikace v souboru `AndroidManifest.xml`.

Ukládání dat pro chod aplikace

Všechny aktivity v rámci implementace využívají tzv. shared preferences, do češtiny může být přeloženo jako sdílené preference. Shared preferences je jedním z mnoha typů ukládání dat aplikace, který ukládá a načítá data ve formě páru klíč-hodnota. Tento typ ukládání je v aplikaci využit především pro načítání hodnot v rámci jiných aktivit, než při kterých jsou stejné hodnoty ukládány. Pro přístup k souborům typu shared preferences slouží třída `SharedPreferencesManagement`. Aplikace pracuje se třemi soubory tohoto typu:

- `auth.xml` - soubor pro ukládání přihlašovacích údajů. Souboru je přidělen parametr `MODE_WORLD_READABLE`, což umožní jeho přečtení jakékoliv aplikaci.
- `settings.xml` - soubor pro ukládání hodnot pro nastavení. Souboru je přidělen parametr `MODE_WORLD_WRITABLE`, což umožní jeho úpravu nejenom dané aplikaci, ale kterékoliv jiné.
- `activation.xml` - soubor pro ukládání hodnot během aktivace. Souboru je také přidělen parametr `MODE_WORLD_READABLE`.

Aktivační aktivita

Aktivační aktivita je první aktivitou při spuštění aplikace. Funkčnost aktivity je implementována v souboru `ActivationActivity.java` a její vzhled v souboru s názvem `activity_activation.xml`. Tato aktivita je označena jako hlavní.

Na aktivační aktivitě musí uživatel vložit telefonní číslo zařízení, na kterém je aplikace spuštěna. Po zadání telefonního čísla uživatel klikne na tlačítko „Potvrdit“. Klik na toto tlačítko zavolá funkci `sendActivationSMS()`, ve které se spustí funkce `generateCode()`, jejíž výstupem je sedmi-místný náhodný kód, který se uloží do souboru `activation.xml`. Dále je volána funkce `sendSMSBroadcast()`, která vytvoří nový intent s názvem akce `SEND_SMS`. Intentu jsou předány hodnoty vloženého telefonního čísla a vygenerovaného kódu. Volaná akce `SEND_SMS` je v manifestu zaregistrovaná jako akce pro zavolání exportovaného broadcast receiveru s názvem `SendSMSReceiver`. Úkolem tohoto receiveru je zaslání SMS na zadané telefonní číslo, jejímž obsahem bude vygenerovaný sedmi-místný kód.

Aby aplikace mohla reagovat na obsah obdržený v SMS, musí dojít k implementaci druhého broadcast receiveru. Tento receiver nese název `ReceiveSMSReceiver` a je vytvořen tak, aby přečetl obsah SMS a porovnal jej s kódem uloženým v souboru `activation.xml`. V případě, že se kód uložený v souboru a kód obdržený v SMS shoduje, je uživatel vpuštěn na přihlašovací aktivitu a aktivace aplikace je dokončena. V opačném případě vpuštěn není a musí aktivaci provést znovu. To, zda je aplikace aktivována či není, je opět uloženo v souboru `activation.xml`. Po úspěšné aktivaci je tato aktivita při dalším spuštění aplikace přeskočena.

Pro odesílání a přijímání SMS zpráv aplikací, jsou v `AndroidManifest.xml` vyžadována povolení `android.permission.SEND_SMS` pro odesílání SMS a povolení `android.permission.RECEIVE_SMS` pro příjímání SMS. Uživatel musí při prvním spuštění aplikace potvrdit tato povolení, bez kterých by aplikace nemohla na tyto akce nijak reagovat.

Účelem této aktivity je demonstrace zranitelnosti exportovaného broadcast receiveru. Exportovaný broadcast receiver `SendSMSReceiver` reaguje na zavolání akce, v tomto případě na zavolání akce `SEND_SMS`. Tato akce je volána pomocí intentu s tzv. extra daty, s jejichž pomocí mohou být data předávána mezi jednotlivými komponentami. Zranitelnost spočívá v tom, že exportovaný broadcast receiver může být lehce odposlechnut. Pokud se v odposlechu objeví nějaká zasílaná extra data, může útočník zneužít tento broadcast receiver k zaslání jiných extra dat. V tomto případě může útočník odeslat SMS zprávu s jakýmkoliv obsahem na jakékoliv telefonní číslo.

Přihlašovací aktivita

Přihlašovací aktivita se objeví po zdárné aktivaci aplikace. Funkčnost aktivity je implementována v souboru `LoginActivity.java` a její vzhled v souboru s názvem `activity_login.xml`.

Tato aktivita obsahuje tři vstupní pole, do kterých uživatel zadává IP adresu WebMailu, login a heslo. Po stisknutí tlačítka „Přihlásit se“ je zavolána funkce `sendRequst()`, která zašle žádost o přihlášení se na server. Využívá se zde API pro přihlášení zmíněné v tab. 4.1. Pokud je žádost zpracována na serveru v pořádku, vrácené hodnoty v odpovědi jsou následně zpracovány aplikací. Vrácené hodnoty `iduser` (ID uživatele), email a jméno souboru profilového obrázku jsou uloženy do souboru `settings.xml`, aby se s nimi mohlo pracovat v dalších požadavcích. Do stejného souboru se dále ukládá IP adresa WebMailu a login, které uživatel zadal do aplikace. Důležitou hodnotou, která se opět ukládá do souboru `settings.xml`, je hodnota pro cookie. Tato hodnota je vygenerována funkcí `generateCode()`, která je použita také v aktivační aktivitě.

V případě, že uživatel zatrhl na přihlašovací obrazovce zaškrtávací pole „Trvalé přihlášení“, ukládají se všechny zadané hodnoty (IP adresa WebMailu, login a heslo) při přihlášení do souboru `auth.xml`. Pokud uživatel aplikaci ukončí a opět ji spustí, je automaticky přihlášen s údaji zaznamenanými v tomto souboru. Pokud se přihlášení nezdaří, například z důvodu nedostupného serveru, je opět zobrazen přihlašovací formulář a uložená data jsou ze souboru `auth.xml` vymazána.

Po úspěšném zpracování požadavku a uložení potřebných hodnot do zmíněných souborů, je uživatel úspěšné přihlášen do aplikace. Tento fakt se projeví přesměrováním uživatele na obrazovku s doručenou poštou. Poté je ještě s 5 sekundovým zpožděním, z důvodu nepřekrývání se jednotlivých volání, ve funkci `volleyDownloadPic()` volán požadavek na stažení profilového obrázku pomocí API popsaném v tab. 4.9. Obrázek je vrácen v odpovědi ve formě bitmapy a je tedy nutné ho uložit v požadovaném formátu. Do funkce `saveImage(Bitmap bitmap)` vstupuje v argumentu bitmapa, která je dále v těle funkce převedena pomocí komprese na požadovaný formát obrázku. Pokud by u názvu profilového obrázku chyběl formát, po kompresi je ve výchozím stavu nastaven na formát JPEG (Joint Photographic Experts Group). Výsledný profilový obrázek je po úspěšné kompresi uložen do vnitřního úložiště aplikace ve složce `app_Images`. Celý proces uložení profilového obrázku proběhne pouze v případě, že uživatel má nějaký profilový obrázek nastaven. Pokud je v `settings.xml` u jména profilového obrázku hodnota `false`, uživatel obrázek nastaven nemá.

Aktivita samotná nemá implementované žádné zranitelnosti, avšak soubor s názvem `auth.xml`, se kterým aktivita pracuje, zranitelný je. Soubor má nastaven již zmíněný parametr `MODE_WORLD_READABLE` a může být tedy přečten jakoukoliv jinou aplikací či komponentou. Tento soubor je tedy možné přečíst skrz content provider, který slouží pro načítání příloh a je zranitelný na Path Traversal.

Aktivita pro zobrazení doručených a odeslaných zpráv

Aktivita jak pro zobrazení doručených zpráv, tak pro zobrazení odeslaných zpráv je stejná a je implementována v souboru `InboxSentActivity.java` a její celkový vzhled v souboru `activity_inbox.java`. Vzhled jednoho řádku (záznamu) je implementován v souboru `row_data_messages_overview.xml`.

Tato aktivita je volána pomocí intentu, kterému jsou předány hodnoty `userId` (ID uživatele) a `folder` (složka) jako extra data. Na základě předané hodnoty složky jsou zobrazeny požadované zprávy uživatele s předaným ID uživatele. Doručené zprávy jsou zobrazeny, pokud je hodnota složky rovna textu `inbox`, a odeslané zprávy se zobrazují po zisku hodnoty složky s textem `sent`.

Po určení složky a uživatele je volána funkce `loadMessages()` pro načtení zpráv. Pro načtení požadovaných zpráv je v rámci funkce voláno API pro načtení zpráv, podle složky, zmíněné v tab. 4.3 nebo v tab. 4.4. V odpovědi je vrácen seznam zpráv, z nichž každá obsahuje stejný seznam parametrů, ale s různým obsahem. Pro lepší manipulaci s obsahem každé zprávy je v aplikaci implementována třída s názvem `MessageInfo`. Při procházení získané odpovědi je tedy obsah každé zprávy zaznamenán do datové struktury `MessageInfo` prostřednictvím konstruktoru. Jednotlivé zprávy jsou dále ukládány do databáze s názvem `MessagesDB`.

Databáze pro ukládání obsahu a početních statistik zpráv je implementována ve třídě `MessagesSQLiteDatabaseHandler`. Databáze `MessagesDB` obsahuje dvě tabulky:

- `messages` - tabulka pro uložení obsahu parametrů jednotlivých zpráv. Obsahuje sloupce, jejichž názvy odpovídají parametrům vrácených v odpovědi API požadavku (tab. 4.3 nebo tab. 4.4). Ke každé zprávě je také ukládáno ID uživatele, se kterým je zpráva spojena, a složka, do které zpráva patří.
- `stats` - tabulka pro uložení počtu doručených a odeslaných zpráv. Obě statistiky jsou opět vždy ukládány s ID uživatele, se kterým jsou statistiky spojeny.

Ve třídě `MessagesSQLiteDatabaseHandler` jsou implementovány funkce pro přidání zprávy, pro přidání statistik, pro zisk všech uložených zpráv, pro zisk všech uložených zpráv konkrétního uživatele a pro odstranění zprávy. Vše pro manipulaci s databází `MessagesDB`.

Zprávy jsou do databáze uloženy pouze v případě, že v ní ještě neexistují. Toto je zajištěno ve funkci `checkIfExistsDB()`, ve které se zjišťuje, zda kombinace ID zprávy, ID uživatele a složky, předaná v argumentu funkce, již v databázi existuje či ne. Tento problém by mohl být vyřešen pomocí určení primárního klíče na ID zprávy, avšak ID zprávy je unikátní pouze v rámci dané složky. Může tedy nastat situace, kdy ID zprávy je stejné pro zprávu v doručené i odeslané poště.

Po zpracování všech zpráv v odpovědi API volání dochází k přidání či aktualizaci počtu statistik. Dále jsou jsou z databáze získány všechny zprávy daného uživatele a složky a následně jsou seřazeny sestupně podle data odeslání. Nakonec jsou získané zprávy zobrazeny uživateli v aplikaci skrz tzv. `ListView`, který umožňuje zobrazit seznam dat v různém rozložení. Z důvodu vlastní implementace vzhledu a chování jednoho řádku (záznamu), byla vytvořena třída s názvem `MessagesAdapter`, jejíž implementace přepisuje systémový vzhled a chování jednoho řádku.

Je nutné zmínit, že obsah databáze se po odhlášení z aplikace nevymazává. Pokud se do aplikace přihlásí několik uživatelů s různými účty, jsou v databázi zaznamenané všechny jejich zprávy.

V této aktivitě jsou implementovány dvě zranitelnosti. První zranitelností je to, že tato aktivita je exportovaná a tudíž může být vyvolána jinou aplikací. Aktivita může být nastartována cizí aplikací pomocí intentu, kterému jsou předána extra data. V tomto případě se intentu předává ID uživatele a složka. Cizí aplikace může tedy zkoušet různá ID a tím neoprávněně získat obsah zpráv ve volané složce. Zranitelnost je možno využít pouze v případě, že je v aplikaci někdo přihlášen.

Druhá zranitelnost je spojena s tabulkou `stats` v databázi `MessagesDB`. Přístup k datům této tabulky je prostřednictvím exportovaného content provideru implementovaného třídou `StatsProvider`. V této třídě se přepisuje primární metoda `query()` pro zisk požadovaných dat. Metodě jsou předávány argumenty `selection` a `protection`, které jsou vytvořeny zřetězením dat. Tento fakt vede k využití zranitelnosti SQL injection. Útočník tak může skrze cizí aplikaci zavolat tento content provider, předat mu hodnoty pro selekci nebo projekci a po odeslání příkazu získat neoprávněný přístup nejen k datům v tabulce `stats`, ale také ke všem datům v databázi `MessagesDB`.

Aktivita pro zobrazení detailu zprávy

Implementace aktivity pro zobrazení detailu zprávy je k dipozici v souboru s názvem `MessageDetailActivity.java`. Celkový vzhled aktivity je pak implementován v souboru `activity_message_detail.xml`. V případě, že zpráva obsahuje přílohy, je vzhled seznamu příloh definován v souboru `row_data_attachments.xml`

K této aktivitě se uživatel může dostat, pokud klikne na konkrétní zprávu zobrazenou v seznamu doručených nebo odeslaných zpráv. Klik na záznam způsobí, že je zavolána funkce `switchToDetail()` ve třídě `MessagesAdapter`. Ve funkci je vytvořen intent, kterému jsou předána všechna data, potřebná pro zobrazení detailu zprávy. Jedná se o jméno, email příjemce nebo email odesílatele, předmět, datum, obsah zprávy, ID zprávy, ID uživatele, seznam příloh a složka. Jakmile je intent poslán, zobrazí se detail zprávy pomocí obsahu vybraných dat.

Uživatel má také možnost v detailu zprávy tuto zprávu smazat. Smazání je provedeno pomocí API pro smazání zprávy popsaném tab. 4.10. Pokud je vrácená odpověď úspěšná, vymaže se daná zpráva také z databáze. Nakonec je uživatel přesměrován zpět na aktivitu, ze které přišel.

V případě, že je součástí zprávy alespoň jedna příloha, je vzhled a chování seznamu příloh definován ve třídě `AttachmentsAdapter`. Tato třída se také stará o to, aby po rozkliknutí detailu zprávy byly všechny přílohy staženy do vnitřního úložiště aplikace a tím vznikla možnost uložené přílohy otevřít.

Stažení obsahu přílohy se provádí zavoláním API zmíněném v tab. 4.5. Toto API je zavoláno tolikrát, kolik je počet příloh ve zprávě. Vzhledem k tomu, že vrácená odpověď může být jakéhokoliv datového typu (obrázek, textový soubor, atd.) a knihovna `volley` neobsahuje pro tento případ předem připravenou třídu, bylo nutné definovat vlastní typ požadavku a odpovědi. Tento problém byl vyřešen ve třídě `CustomVolleyDownloadAttachments`. Po obdržení odpovědi je finální výstup sestaven do požadované podoby a uložen do adresáře `files` ve funkci `saveFile()`.

Zranitelnost v rámci této aktivity najdeme u špatné implementace content provideru, který slouží pro přístup k přílohám. Tento content provider je exportovaný a je definován ve třídě `FileProvider`, ve které je přepisována metoda pro otevření souboru `openFile()`. Metoda neověřuje příchozí URI, což může být zneužito zranitelností Path Traversal. Útočník se tak může dostat nejen ke konkrétnímu souboru v příloze, ale také k ostatním souborům mimo vnitřního úložiště aplikace.

K obsahu zprávy se útočník může také dostat při změně ID zprávy během debuggingu.

Aktivita pro zobrazení kontaktů

Zobrazení kontaktů je implementováno v souboru `ContactsActivity.java`. Celkový vzhled této aktivity je pak určen v souboru `activity_contacts.xml`. Vzhled jednoho záznamu je definován v souboru `row_data_contacts.xml`.

Spuštění aktivity vede k zavolání API pro zobrazení kontaktů zmíněného v tab. 4.8. V odpovědi je vrácen seznam kontaktů, kde každý záznam obsahuje ID kontaktu, jméno, email a popis. Manipulace s obsahem jednotlivých kontaktů je snadnější

díky třídě `ContactsInfo`. Obsah kontaktů je tak ukládán do vlastní datové struktury `ContactsInfo` pomocí konstruktoru. Každý kontakt je ukládán do databáze s názvem `ContactsDB`.

Třída `ContactsSQLiteDatabaseHandler` implementuje všechny funkce databáze `ContactsDB`. Tato databáze obsahuje jednu tabulka s názvem `contacts`, ve které jsou uloženy obsahy parametrů jednotlivých kontaktů. Do tabulky se zaznamenává nejen obsah parametrů vrácených v odpovědi, ale také se ke každému záznamu navíc přidává ID uživatele, který kontakty vlastní.

Ve třídě `ContactsSQLiteDatabaseHandler` jsou dále definovány metody pro přidání kontaktu, pro zisk všech kontaktů, pro zisk kontaktů konkrétního uživatele, pro odstranění kontaktu a pro vyhledání kontaktu. Pro srovnání lze říci, že implementace databáze `ContactsDB` a výše zmíněné databáze `MessagesDB` je založena na stejném postupu.

Kontakty jsou do databáze uloženy pouze v případě, že v ní ještě neexistují, tak jak tomu je i u zpráv. V tomto případě je pro zjištění přítomnosti kontaktu v databázi využita unikátnost ID kontaktu. ID kontaktu je tedy v kódu aplikace zvoleno jako primární klíč.

Po uložení kontaktů do databáze dojde k abecednímu seřazení kontaktů. Poté se kontakty zobrazí uživateli prostřednictvím již zmíněného `ListView`. Opět byla nutná vlastní implementace chování a vzhledu jednoho záznamu. Tato nutnost byla vyřešena ve třídě `ContactsAdapter`. Navíc je v této třídě definováno vymazání kontaktu. Kontakt je vymazán po úspěšném zavolání API pro smazání kontaktu popsaném v tab. 4.10. Kontakt je také vymazán z databáze.

Tato aktivita dále umožňuje vyhledávání kontaktů. Vyhledávání je umožněno přes vyhledávací okno, do kterého se vepisuje hledané jméno. Požadovaný kontakt je zobrazen pouze ve chvíli, kdy se vložený text přesně shoduje se jménem kontaktu uloženém v databázi. Uživateli tedy nestačí napsat pouze počáteční písmeno či jenom část jména.

Stejně jako v případě databáze `MessagesDB`, obsah databáze `ContactsDB` není smazán po odhlášení uživatele. V databázi jsou tak uloženy všechny kontakty těch uživatelů, kteří se dříve do aplikace přihlásili.

V této aktivitě jsou také záměrně implementované chyby, které slouží k demonstraci zranitelností. První zranitelnost je SQL injection, které lze dosáhnout skrz špatnou implementaci funkce pro vyhledávání kontaktů. Když uživatel zadá text do vyhledávacího pole a potvrdí ho, tento text je na přímo vložen do SQL příkazu. Pokud útočník zvolí vstupní text, který způsobí vyhodnocení celého SQL příkazu jako `true` (pravda), pak aplikace zobrazí všechny kontakty uložené v databázi - tedy i ty, které nepatří přihlášenému uživateli.

Druhá zranitelnost demonstruje špatnou definici oprávnění exportovaného content provideru pro přístup k obsahu tabulky `contacts`. Tento contento content provider je definován ve třídě `ContactsProvider`. Obsah tabulky je poskytnut v případě, že konec URI cesty obsahuje `/contacts` nebo `/contacts/`. V manifestu je pro tento content provider zapsáno, že pro přístup k obsahu s cestou `/contacts` je vyžadováno oprávnění pro čtení. Nicméně útočníkovi stačí, aby za cestu zapsal lomítko a tedy volal cestu `/contacts/`. Tato cesta již oprávnění nepotřebuje a útočníkovi je zobrazen celý obsah tabulky `contacts`. Navíc útočník může skrze stejnou URI cestu přidávat nové kontakty, protože právo pro zápis není vyžadováno.

Aktivita pro přidání nového kontaktu

V souboru `NewContactActivity.java` je implementována aktivita pro přidání nového kontaktu a v souboru `activity_new_contact.xml` její vzhled.

Aktivita je spuštěna ve chvíli, kdy uživatel klikne na tlačítko „Přidat kontakt“ na obrazovce pro zobrazení kontaktů. Uživatel dále pokračuje vyplněním jména, emailu a popisu nového kontaktu. Po stisknutí tlačítka „Uložit kontakt“ je provedena funkce `volleyAddContact()`, ve které je zavoláno API pro přidáno nového kontaktu definovaného v tab. 4.7.

Přidat kontakt lze pouze pokud má uživatel aktivovanou plnou verzi aplikace. Plná verze nelze nijak aktivovat přes nastavení, a tak je na uživateli, zda zkusí změnit verzi jiným způsobem. Verze aplikace je zaznamenána v souboru `settings.xml`. Tento soubor má přidělen parametr pro zápis, a tak může být jednoduše změněn. Modifikací uložené hodnoty `appVersion` z `trial` na `full`, dosáhne uživatel aktivování plné verze aplikace. Kontrolu, zda se jedná o plnou verzi bude také možné obejít debuggingem.

Aktivita pro zaslání nové zprávy

Důležitou součástí aplikace je aktivita pro zasílání zpráv, která je definována v souboru `NewMessageActivity.java`. Vzhled této aktivity je zase implementován v souboru `activity_new_message.xml`.

Na této aktivitě uživatel vyplní příjemce, předmět a text zprávy a celou zprávu zašle po stisknutí tlačítka „Odeslat zprávu“. V případě, že je zadán alespoň příjemce a předmět, je příkaz k odeslání zprávy obsloužen funkcí `volleySendMessage()`. V této funkci je voláno API pro zaslání nové zprávy, které je popsáno v tab. 4.6. Pokud jsou zmíněná pole správně vyplněna, je zpráva úspěšně odeslána. Poté je uživatel přesměrován na obrazovku s doručenou poštou.

Tato aktivita je exportovaná a může být tak vyvolána i externě bez přihlášení. Jiné zranitelnosti nejsou v této aktivitě implementované.

Aktivita pro zobrazení nastavení

Poslední aktivitou je aktivita pro zobrazení nastavení, která je implementována v souboru `SettingsActivity.java` a její vzhled v souboru `activity_settings.xml`.

Aktivita slouží pouze pro lepší přehlednost použitých hodnot v aplikaci. Všechny hodnoty jsou získány ze souboru `settings.xml`.

Ostatní třídy

Mezi ostatní třídy, které nebyly použity ve zmíněných aktivitách, patří třída pro implementaci hlavní nabídky, odhlášení a oznámení nově příchozí zprávy.

K hlavní nabídce se lze dostat skrze aktivity, které jsou k dipozici po příhlášení. Hlavní nabídka je implementována ve třídě `Navigation`. Celkový vzhled nabídky je rozdělen na dvě části. V první části, v souboru `drawer_menu.xml`, jsou určeny ikony a názvy jednotlivých položek nabídky. Ve druhé části, v souboru `nav_header.xml`, je definována hlavička hlavní nabídky. Ve třídě `Navigation` jsou spojeny obě části tak, aby vznikla plně funkční zajížděcí nabídka. V hlavičce hlavní nabídky se nastavuje profilový obrázek, verze aplikace, login a email uživatele z již získaných hodnot uložených v souboru `settings.xml`.

O detekci nově příchozí zprávy se stará služba, která je implementovaná ve třídě `ListenNewMessageService`. Tato služba běží na pozadí a má za úkol se každých 30 sekund ptát, zda nedošla nová zpráva. V nastalém časovém intervalu služba zavolá API pro získání doručených zpráv. Ze získané odpovědi zjistí počet doručených zpráv a tento počet porovná s počtem doručených zpráv, které jsou již uloženy v databázi. Pokud je číslo ze získané odpovědi vyšší než číslo z databáze, znamená to, že uživatel obdržel novou zprávu. V ten moment služba aktivuje pomocí akce `SEND_NEW_MESSAGE` brodcast receiver, který je definován ve třídě `SendNewMessagesInfoReceiver` a kterému předá parametry nově příchozí zprávy - email odesílatele a obsah zprávy. Brodcast receiver tak informuje uživatele o obdržení nové zprávy pomocí zobrazení oznámení, ve kterém je uveden email odesílatele a obsah nově doručené zprávy.

Zmíněný broadcast receiver i zmíněná služba jsou exportované a tudíž mohou být zranitelní. Útočník je v první řadě schopen odposlechnout data, která jsou zasílána ze služby do broadcast receiveru. Útočník tak může zjistit důvěrný obsah zasílaných zpráv. Dále může útočník zneužít brodcast receiver k zobrazení oznámení na zařízení uživatele s jakýmkoliv obsahem.

Poslední ještě nezmíněná třída aplikace je třída `Logout` pro odhlášení uživatele z aplikace. Odhlášení nevyžaduje žádný vzhled obrazovky, a proto tato akce nebyla implmentována jako aktivita. Uživatel se může odhlásit přes hlavní nabídku po

kliknutí na položku „Odhlásit se“. Klik způsobí zavolání API pro odhlášení se zmíněné v tab. 4.2. Pokud nedošlo k problémové situaci, uživatel je odhlášen a všechny zaznamenané hodnoty pro běh aplikace jsou ze souborů vymazány.

4.5 Testování aplikace

Testování aplikace bylo prováděno již během vývoje, aby se zamezilo vzniku chyb zasahujících do následné implementace nových částí. Aplikace byla vyvíjena i testována na emulátoru Pixel 2 s Android verzí 10. Na stejném emulátoru byla také otestována finální verze aplikace, která již byla sestavená do spustitelného souboru `PenterepMail.apk`. Aplikace byla dále otestována na fyzickém mobilním zařízení Huawei Nova 3 s Android verzí 9 (API 28). Během finálního testování výše zmíněných funkčností nebyla nalezena žádná chyba. Nicméně je velmi pravděpodobné, že aplikace nějaké menší nedokonalosti stále obsahuje.

Příprava testování zranitelností

Po otestování funkčnosti aplikace byly otestovány, a tedy i demonstrovány, záměrně implementované zranitelnosti. Pro zneužití zranitelností byl použit nástroj Drozer ve verzi 2.4.4. Jedná se o framework vytvořený v programovacím jazyce Python a určený pro systém Android, který pomáhá vývojářům otestovat vytvořenou aplikaci z hlediska její bezpečnosti. Drozer obsahuje komplexní funkce, které poskytují možnost vyzkoušet si zneužít veřejně dostupné zranitelnosti systému Android. Detailnější informace o nástroji Drozer včetně možnosti jeho stažení jsou k dispozici na stránce `https://labs.f-secure.com/tools/drozer/`.

Pro demonstraci zranitelností vytvořené aplikace bylo zapotřebí stažení nástroje Drozer ve dvou částech. První část byla nainstalována do osobního počítače a ovládá se pomocí příkazového řádku. Druhá část představuje klientského agenta (mobilní aplikaci) a byla uvedena do chodu na emulátoru. Tyto dvě části bylo dále nutno propojit. U první části nástroje muselo proběhnout přes příkazový řádek přesměrování portu tak, aby byl notebooku napojen na TCP soket otevřený agentem. Ve výchozím nastavení nástroje je použit port 31415. Dále byl spuštěn agent na emulátoru a v něm byl zapnut vestavěný server. Samotné propojení obou částí proběhlo opět přes příkazový řádek [86].

Popsané akce byly provedeny příkazy 4.1 a 4.2 . Příkaz 4.1 bylo nutné spustit ve složce, ve které je uložen používaný emulátor. U příkazu 4.2 je nástroj spouštěn ze složky, ve které je nainstalován.

Výpis 4.1: Přesměrování portu

```
"C:\AndroidStudio\platform-tools␣adb␣forward␣tcp:31415␣
   tcp:31415"
```

Výpis 4.2: Propojení obou částí nástroje Drozer

```
"C:\Python27\Scripts␣drozer␣console␣connect"
```

Demonstrace vybraných zranitelností

Připravené prostředí pro testování pomocí nástroje Drozer bylo využito pro demonstraci vybraných zranitelností. První ukázkou je zranitelnost broadcast receiveru implementovaného v aktivační aktivitě. V první části obr. 4.4 je zobrazen příkaz, po jehož potvrzení začne nástroj Drozer odposlouchávat komunikaci vyvolanou akcí `SEND_SMS`. Dále ve spodní části stejného obrázku je vidět, že odposlech zmíněné akce byl úspěšný. Útočník, který akci odposlouchává, tak získal nejen telefonní číslo, na které byl aktivační kód zaslán, ale také samotný obsah zprávy. Na obr. 4.5 je pak zachycena zpráva, která byla odposlechnuta.

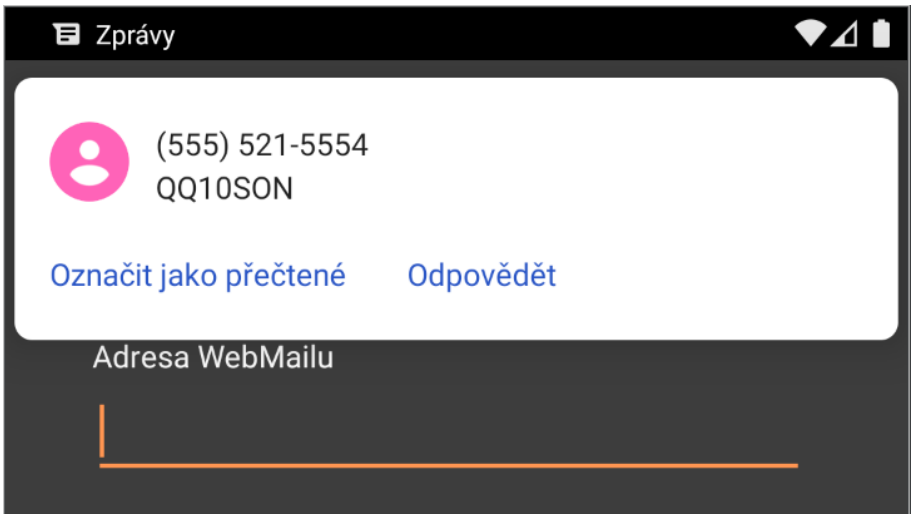
```
dz> run app.broadcast.sniff --action SEND_SMS
[*] Broadcast receiver registered to sniff matching intents
[*] Output is updated once a second. Press Control+C to exit.

Action: SEND_SMS
Raw: Intent { act=SEND_SMS flg=0x10 (has extras) }
Extra: phoneNumber=5554 (java.lang.String)
Extra: message=QQ10SON (java.lang.String)
```

Obr. 4.4: Ukázka odposlechu broadcast receiveru.

Další vybranou zranitelností je zranitelnost SQL Injection, která byla záměrně vložena do aktivity pro zobrazení kontaktů. Prerekvizita tohoto útoků spočívala v tom, že se do aplikace na stejném zařízení přihlásili dva uživatelé, jejichž kontakty byly uloženy do lokální databáze. Na obr. 4.6 jsou ukázány kontakty uživatele A. Uživatel A dále zadal do vyhledávacího okna výraz pro zneužití zranitelnosti SQL Injection. Výsledek je vyobrazen na obr. 4.7, na kterém je vidět, že se uživateli A zobrazily nejen jeho kontakty, ale také ostatní kontakty uložené v databázi.

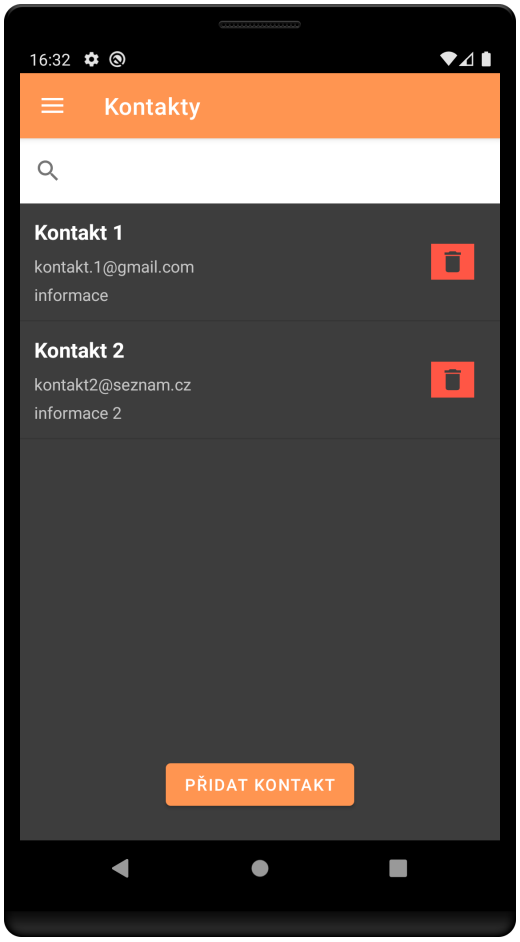
Zranitelnost Path Traversal je další zranitelností vybranou pro ukázku. Tato zranitelnost je implementovaná v aktivitě pro zobrazení detailu zprávy a souvisí se špatným zavedením content provideru v kódu a s chybně zvoleným parametrem souboru `auth.xml`, který umožňuje jeho přečtení jakoukoli jinou komponentou či



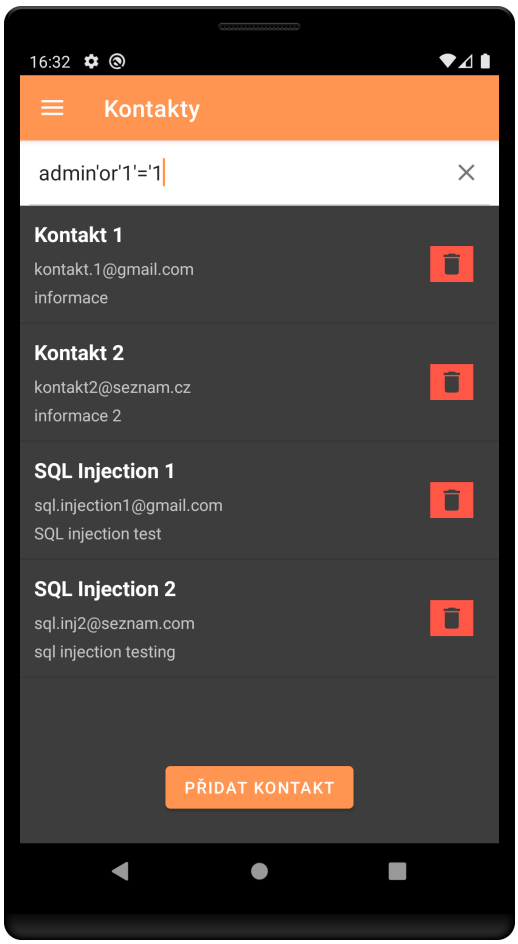
Obr. 4.5: Zaslaná SMS získaná odposlechem.

aplikací. První příkaz na obr. 4.8 umožňuje přečtení obsahu souboru, který byl zaslán jako příloha. Konkrétně se jedná o soubor s názvem `pokus.txt`. Druhý příkaz na obrázku slouží ke stejnému účelu, avšak je pomocí content provideru přistoupeno k souboru, který leží v úplně jiné složce. Pomocí zneužití zranitelnosti Path Traversal se tak útočník dostal k obsahu souboru `auth.xml`, ve kterém zjistil přihlašovací údaje přihlášeného uživatele včetně jeho hesla.

Poslední vybranou zranitelností pro ukázku je zranitelnost služby a brodcast receiveru, která je implementovaná ve třídě pro detekci příchozí zprávy. Na obr. 4.9 je vidět příkaz pro zaslání extra dat konkrétní akci. V tomto případě se zasílá obsah „email“ a „zprava“ akci `SEND_NEW_MESSAGE`. Výsledek je ukázán na obr. 4.10, kde se na zařízení s aplikací objevilo oznámení včetně zaslaného obsahu.



Obr. 4.6: Kontakty přihlášeného uživatele.



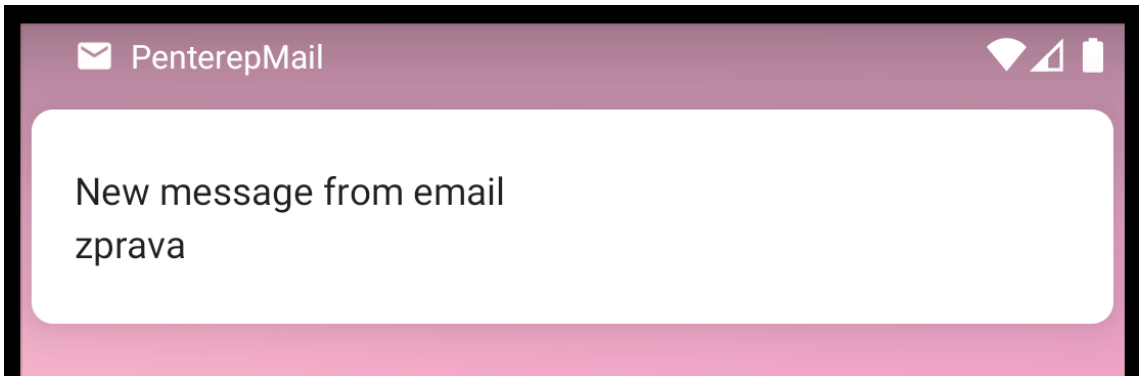
Obr. 4.7: Využití zranitelnosti SQL Injection.

```
dz> run app.provider.read content://com.example.penterepmailmobileapp.fileprovider
/pokus.txt
testovaci textovy soubor
dz> run app.provider.read content://com.example.penterepmailmobileapp.fileprovider
/../../../../data/data/com.example.penterepmailmobileapp/shared_prefs/auth.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="addressKey">10.0.0.26</string>
    <string name="loginKey">kaja</string>
    <string name="passwordKey">12345</string>
</map>
```

Obr. 4.8: Využití zranitelnosti Path Traversal.

```
dz> run app.broadcast.send --action SEND_NEW_MESSAGE --extra string from "email" -
-extra string message "zprava"
```

Obr. 4.9: Ukázka zaslání dat dané akci.



Obr. 4.10: Zobrazené oznámení pomocí zavolání dané akce.

Závěr

Tato práce byla rozdělena do čtyř hlavních kapitol. První část se věnovala bezpečnosti mobilních aplikací včetně představení hlavních mobilních operačních systémů, popsání současného stavu bezpečnosti a organizace OWASP. V rámci této organizace byly uvedeny dva projekty zabývající se deseti největšími bezpečnostními riziky mobilních aplikací a testováním mobilních aplikací.

Druhá část se zaměřovala na vývoj mobilní aplikace pro systém Android, kde byla popsána architektura tohoto systému a dále používaná vývojová prostředí, nástroje a programovací jazyky. Také zde byly zmíněny základní bloky aplikace.

Třetí kapitola byla zaměřena na vlastní návrh aplikace. Byla zde provedena analýza zranitelností, které byly dále využity pro implementaci. Také byl vytvořen návrh a vzhled základních bloků aplikace. V poslední části třetí kapitoly byly vybrány vhodné nástroje a technologie pro vývoj mobilní aplikace.

Ve čtvrté kapitole byla vysvětlena vlastní implementace mobilní aplikace. V první části kapitoly byla popsána příprava prostředí. Poté zde byla objasněna struktura aplikace a dále zde bylo shrnuto využití API volání. Hlavní chování aplikace bylo ujasněno v části Implementace jednotlivých aktivit a tříd. Nakonec byla aplikace řádně otestována a byly zde také ukázány vybrané zranitelnosti.

Všechny stanovené cíle diplomové práce byly splněny. V rámci diplomové práce byla nastudována problematika bezpečnosti mobilních aplikací a požadavků na jejich vývoj pro systém Android. Dále byla aplikace navržena, implementována a otestována. Aplikace tak může být považována za plně funkční.

Článek na téma této diplomové práce byl také oceněn druhým místem v soutěži EEICT (Electrical Engineering, Information and Communication Technologies) pořádané Fakultou elektrotechniky a komunikačních technologií Vysokého učení technického v Brně.

Aplikace bude dále volně dostupná a bude sloužit především ke vzdělávání v souvislosti s kybernetickou bezpečností, čímž pomůže zvýšit úroveň kybernetické bezpečnosti v České republice.

Literatura

[1] *McAfee Mobile Threat Report* [online]. McAfee, 2020 [cit. 14. 11. 2020]. Dostupné z: <https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf>.

[2] ADOLPH, M. Mobile Applications. *ITU News*. 2009, č. 6, ISSN 1020-4148.

[3] BURDA, K. *Bezpečnost informačních systémů*. Brno: Vysoké učení technické v Brně, 2013. ISBN 978-80-214-4890-2.

[4] *Žebříčky nejlepších* [online]. Google Play, 2020 [cit. 15. 10 2020]. Dostupné z: <`https://play.google.com/store/apps/top`>.

[5] *State of Mobile 2020* [online]. App Annie, 2020 [cit. 15. 10. 2020]. Dostupné z: <`https://go.appannie.com/rs/071-QED-284/images/2001_State_of_Mobile_2020_Main_EN.pdf`>.

[6] VACCA, J. *Computer and Information Security Handbook*. San Francisco: Morgan Kaufmann Publishers Inc., 2017. ISBN 978-0-12-803843-7.

[7] KODYM, O. *Operační systémy*. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2013. ISBN 978-80-248-3053-7.

[8] *What are the different mobile operating systems?* [online]. Uswitch, 2016 [cit. 15. 10. 2020]. Dostupné z: <`https://www.uswitch.com/mobiles/guides/mobile-operating-systems/`>.

[9] *Mobile Operating System Market Share Worldwide* [online]. StatCounter, 2020 [cit. 15. 10. 2020]. Dostupné z: <`https://gs.statcounter.com/os-market-share/mobile/worldwide`>.

[10] *Platforma* [online]. Alza.cz, 2020 [cit. 17. 10. 2020]. Dostupné z: <`https://www.alza.cz/slovnik/platforma`>.

[11] *Set up for Android Development* [online]. Source Android, 2020 [cit. 1. 11. 2020]. Dostupné z: <`https://source.android.com/setup`>.

[12] MISHRA, P. *Top 20 Tools for Android Development* [online]. 10. 6. 2020 [cit. 1. 11. 2020]. Dostupné z: <`https://mindmajix.com/iot-devices`>.

[13] *Create an Android project* [online]. Google Developers, 2020 [cit. 1. 11. 2020]. Dostupné z: <`https://developer.android.com/training/basics/firstapp/creating-project`>.

[14] *How can we help you?* [online]. Google Help, 2020 [cit. 17. 10. 2020]. Dostupné z: <https://support.google.com/googleplay/android-developer/?hl=en#topic=7071529>.

[15] *Zajištění bezpečného a důvěryhodného prostředí pro všechny* [online]. Google, 2020 [cit. 17. 10. 2020]. Dostupné z: <https://play.google.com/about/developer-content-policy/>.

[16] HINDY, J. *How to install third party apps without the Google Play Store* [online]. 17. 8. 2010 [cit. 18. 10. 2020]. Dostupné z: <https://www.androidauthority.com/how-to-install-apks-31494/>.

[17] AHVANOOEY. M. T.; LI, Q.; RABBANI, M.; RAJPUT, A. R. International Journal of Advanced Computer Science and Applications. *A Survey on Smartphones – Security: Software Vulnerabilities, Malware, and Attacks*, 2017, sv. 8, č. 10. ISSN 2156-5570.

[18] *Hrozby pro Android v dubnu: Češi se nejčastěji potýkali s trojskými koňmi, stahují si je do zařízení sami* [online]. 25. 5. 2020 [cit. 1. 11. 2020]. Dostupné z: <https://www.eset.com/cz/o-nas/pro-novinare/tiskove-zpravy/hrozby-pro-android-v-dubnu-cesi-se-nejcasteji-potykali-s-trojskymi-konmi-stahuji-si-je-do-zarizeni/>.

[19] KENTON, W. *Apple iOS* [online]. 21. 7. 2020 [cit. 3. 11. 2020]. Dostupné z: <https://www.investopedia.com/terms/a/apple-ios.asp>.

[20] *iOS* [online]. 28. 8. 2020 [cit. 3. 11. 2020]. Dostupné z: <https://www.techopedia.com/definition/25206/ios>.

[21] *What's The Best Language To Learn To Build IOS Apps* [online].[cit. 3. 11. 2020]. Dostupné z: <https://buildfire.com/programming-languages-ios-app/>.

[22] *App Review* [online]. iOS Developer, 2020 [cit. 3. 11. 2020]. Dostupné z: <https://developer.apple.com/app-store/review/>.

[23] IMMELT, W. *How to find Third-Party Apps on iOS* [online]. 27. 11. 2019 [cit. 3. 11. 2020]. Dostupné z: <https://www.tmcnet.com/topics/articles/2019/11/27/443870-how-find-third-party-apps-ios.htm>.

[24] *Apple iPhone at risk of hacking through email app* [online]. BBC, 24. 4. 2020 [cit. 3. 11. 2020]. Dostupné z: <https://www.bbc.com/news/technology-52391759>.

[25] LAZARESKA. L.; JAKIMOSKI, K. International Journal of Engineering Management. *Analysis of the Advantages and Disadvantages of Android and iOS Systems and Converting Applications from Android to iOS Platform and Vice Versa*, 2017, sv. 6, č. 5. ISSN 2327-2473.

[26] HILL, S.; JANSEN, M. *Android vs. iOS: Which smartphone platform is the best?* [online]. 24. 10. 2020 [cit. 4. 11. 2020]. Dostupné z: <`https://www.digitaltrends.com/mobile/android-vs-ios/`>.

[27] CHACHAL. D.; SINGH, A; KUMAR, A. International Journal of Research in Engineering, Science and Management . *Security Holes: An Insight of Mobile Operating System*, 2020, sv. 3, č. 4. ISSN 2581-5792.

[28] *Who is the OWASP Foundation?* [online]. OWASP Foundation, Inc., 2020 [cit. 4. 11. 2020]. Dostupné z: <`https://owasp.org/`>.

[29] *OWASP mobile security* [online]. OWASP Foundation, Inc., 2020 [cit. 4. 11. 2020]. Dostupné z: <`https://owasp.org/www-project-mobile-security/`>.

[30] *OWASP Mobile Top 10* [online]. OWASP Foundation, Inc., 2020 [cit. 4. 11. 2020]. Dostupné z: <`https://owasp.org/www-project-mobile-top-10/`>.

[31] *OWASP Mobile Security Testing Guide* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <`https://owasp.org/www-project-mobile-security-testing-guide/`>.

[32] BASATWAR, G. *OWASP Mobile Top 10: A Comprehensive Guide For Mobile Developers To Counter Risks* [online]. 23. 1. 2020 [cit. 8. 11. 2020]. Dostupné z: <https://www.appsealing.com/owasp-mobile-top-10-a-comprehensive-guide-for-mobile-developers-to-counter-risks//>.

[33] *M1: Improper Platform Usage* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <`https://owasp.org/www-project-mobile-top-10/2016-risks/m1-improper-platform-usage`>.

[34] *M2: Insecure Data Storage* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <`https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage`>.

[35] *M3: Insecure Communication* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication>.

[36] *M4: Insecure Authentication* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m4-insecure-authentication>.

[37] *M5: Insufficient Cryptography* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m5-insufficient-cryptography>.

[38] *M6: Insecure Authorization* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m6-insecure-authorization>.

[39] *M7: Poor Code Quality* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m7-client-code-quality>.

[40] *M8: Code Tampering* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m8-code-tampering>.

[41] *M9: Reverse Engineering* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m9-reverse-engineering>.

[42] *M10: Extraneous Functionality* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://owasp.org/www-project-mobile-top-10/2016-risks/m10-extraneous-functionality>.

[43] MUELLER, B.; SCHLEIER, S.; WILLEMSEN, J. *Mobile Security Testing Guide*. OWASP Foundation, Inc., 2019. ISBN 978-0-359-47489-9.

[44] MUELLER, B.; SCHLEIER, S. *Mobile Application Security Verification Standard*. [online]. OWASP Foundation, Inc., 2018 [cit. 8. 10. 2020]. Dostupné z: <https://mobile-security.gitbook.io/masvs/>.

[45] *Checklists* [online]. OWASP Foundation, Inc., 2020 [cit. 8. 11. 2020]. Dostupné z: <https://github.com/OWASP/owasp-mstg/tree/master/Checklists>.

[46] *Understanding the App Development Life Cycle* [online]. 10. 5. 2017 [cit. 9. 11. 2020]. Dostupné z: <https://devops.com/understanding-app-development-life-cycle/>.

[47] *System and kernel security* [online]. Source Android, 2020 [cit. 9. 11. 2020]. Dostupné z: <https://source.android.com/security/overview/kernel-security>.

[48] *Platform Architecture* [online]. Google Developers, 2020 [cit. 9. 11. 2020]. Dostupné z: <https://developer.android.com/guide/platform>.

[49] HARVEY, C. *Top Android IDEs for Developers* [online]. 15. 9. 2017 [cit. 9. 11. 2020]. Dostupné z: <https://www.developer.com/ws/android/development-tools/top-android-ides-for-developers.html>.

[50] HARKIRAN78. *Top Programming Languages for Android App Development* [online]. 19. 5. 2019 [cit. 9. 11. 2020]. Dostupné z: <https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/>.

[51] STONE, S. *Top 20 Tools for Android Development* [online]. 4. 5. 2018 [cit. 9. 11. 2020]. Dostupné z: <https://www.altexsoft.com/blog/engineering/top-20-tools-for-android-development/>.

[52] *Android Studio* [online]. Google Developers, 2020 [cit. 9. 11. 2020]. Dostupné z: <https://developer.android.com/studio>.

[53] *Create an Android project* [online]. Google Developers, 2020 [cit. 9. 11. 2020]. Dostupné z: <https://developer.android.com/training/basics/firstapp/creating-project>.

[54] *Add C and C++ code to your project* [online]. Google Developers, 2020 [cit. 9. 11. 2020]. Dostupné z: <https://developer.android.com/studio/projects/add-native-code>.

[55] SHANKAR, R. *What is Java?* [online]. 16. 9. 2020 [cit. 9. 11. 2020]. Dostupné z: <https://hackr.io/blog/what-is-java>.

[56] *Develop Android apps with Kotlin* [online]. Google Developers, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://developer.android.com/kotlin>.

[57] VAATI, E. *What Is the Android SDK and How to Start Using It* [online]. 2. 7. 2020 [cit. 11. 11. 2020]. Dostupné z: <https://code.tutsplus.com/tutorials/the-android-sdk-tutorial--cms-34623>.

[58] *Android versions comparison* [online]. SocialCompare, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://socialcompare.com/en/comparison/android-versions-comparison>.

[59] *Codenames, Tags, and Build Numbers* [online]. Source Android, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://source.android.com/setup/start/build-numbers>.

[60] *Application security* [online]. Source Android, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://source.android.com/security/overview/app-security>.

[61] *Introduction to Activities* [online]. Google Developers, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://developer.android.com/guide/components/activities/intro-activities>.

[62] *Services overview* [online]. Google Developers, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://developer.android.com/guide/components/services>.

[63] KOTIPALLI, S. R.; IMRAN, M. A. *Hacking Android*. Birmingham: Packt Publishing Ltd., 2016. ISBN 978-1-78588-314-9.

[64] *ContentProvider* [online]. Google Developers, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://developer.android.com/reference/android/content/ContentProvider.html>.

[65] *Intent* [online]. Google Developers, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://developer.android.com/reference/android/content/Intent.html>.

[66] *App Manifest Overview* [online]. Google Developers, 2020 [cit. 11. 11. 2020]. Dostupné z: <https://developer.android.com/guide/topics/manifest/manifest-intro>.

[67] *<activity>* [online]. Google Developers, 2020 [cit. 14. 11. 2020]. Dostupné z: <https://developer.android.com/guide/topics/manifest/activity-element#exported>.

[68] *CWE-926: Improper Export of Android Application Components* [online]. CWE, 2020 [cit. 14. 11. 2020]. Dostupné z: <https://cwe.mitre.org/data/definitions/926.html>.

[69] NIDECKI, T. A. *What Are Insecure Direct Object References* [online]. 23. 3. 2010 [cit. 14. 11. 2020]. Dostupné z: <https://www.acunetix.com/blog/web-security-zone/what-are-insecure-direct-object-references/>.

[70] *Enumeration* [online]. Xperience 13 Documentation [cit. 14. 11. 2020]. Dostupné z: <https://docs.xperience.io/securing-websites/developing-secure-websites/enumeration>.

[71] AHMED, M. I; HASSAN, M. M; BHUYIAN T. Journal of Physics: Conference Series *Local File Disclosure Vulnerability: A Case Study of Public-Sector Web Applications*, 2017, sv. 933. ISBN: 1742-6596.

[72] JACOB, J. *Reflected File Download(RFD) Vulnerability. What? How?* [online]. 14. 6. 2018 [cit. 1. 5. 2021]. Dostupné z: <https://medium.com/@Johne_Jacob/rfd-reflected-file-download-what-how-6d0e6fdbe331>.

[73] BISWAS. S.; SAJAL, M. M. H. K; AFRIN, T; BHUYIAN, T; HASSAN, M. M. International Conference on Cyber Security and Computer Science *A Study on Remote Code Execution Vulnerability in Web Applications*, 2018. s. 50-57. ISBN:978-605-9554-32-9.

[74] *Security tips* [online]. Google Developers, 2020 [cit. 15. 11. 2020]. Dostupné z: <https://developer.android.com/training/articles/security-tips>.

[75] *<application>* [online]. Google Developers, 2020 [cit. 15. 11. 2020]. Dostupné z: <https://developer.android.com/guide/topics/manifest/application-element>.

[76] SRINIVAS. *Android Hacking and Security, Part 3: Exploiting Broadcast Receivers* [online]. 23. 4. 2014 [cit. 15. 11. 2020]. Dostupné z: <https://resources.infosecinstitute.com/topic/android-hacking-security-part-3-exploiting-broadcast-receivers/>.

[77] *Data and file storage overview* [online]. Google Developers, 2020 [cit. 15. 11. 2020]. Dostupné z: <https://developer.android.com/training/data-storage#filesInternal>.

[78] CORNELL, D. *Using Static Analysis to Review File Access in Android Apps* [online]. 20. 4. 2011 [cit. 15. 11. 2020]. Dostupné z: <https://www.denimgroup.com/resources/blog/2011/04/using-static-analysis-to-review-file-access-in-android-apps/>.

[79] *SQL Injection* [online]. OWASP Foundation Inc., 2020 [cit. 15. 11. 2020]. Dostupné z: <https://owasp.org/www-community/attacks/SQL_Injection>.

[80] *Full Path Disclosure* [online]. OWASP Foundation, Inc., 2021 [cit. 1. 5. 2021]. Dostupné z: <https://owasp.org/www-community/attacks/Full_Path_Disclosure>.

[81] *Content providers* [online]. Google Developers, 2020 [cit. 15. 11. 2020]. Dostupné z: <https://developer.android.com/guide/topics/providers/content-providers>.

[82] *Path Traversal* [online]. OWASP Foundation Inc., 2020 [cit. 15. 11. 2020]. Dostupné z: <https://owasp.org/www-community/attacks/Path_Traversal>.

[83] *Path Traversal Vulnerability* [online]. Google Help, 2020 [cit. 15. 11. 2020]. Dostupné z: <https://support.google.com/faqs/answer/7496913?hl=en>.

[84] *Penterepmail* [online]. HACKER Consulting s.r.o., 2020 [cit. 20. 11. 2020]. Dostupné z: <https://www.penterep.com/penterepmail>.

[85] *Volley overview* [online]. Google Developers, 2020 [cit. 1. 5. 2021]. Dostupné z: <https://developer.android.com/training/volley>.

[86] *drozer - User Guide* [online]. MWR InfoSecurity, 2015 [cit. 1. 5. 2021]. Dostupné z: <https://labs.f-secure.com/assets/BlogFiles/mwri-drozer-user-guide-2015-03-23.pdf>.

Seznam symbolů, veličin a zkratek

3G	Third Generation
API	Application Programming Intterface
APK	Android Application Package
DEX	Dalvik Executable
DHCP	Dynamic Host Configuration Protocol
EEICT	Electrical Engineering, Information and Communication Technologies
ES	Embedded Systems
FPD	Full Path Disclosure
GSM	Global System for Mobile Communications
ID	Identification
IDE	Integrated Development Environment
IDOR	Insecure Direct Object Reference
iOS	iPhone Operating System
IP	Internet Protocol
L1	Level 1
L2	Level 2
LFD	Local File Disclosure
MASVS	Mobile Application Security Verification Standard
MiTM	Man in the Middle
MSTG	Mobile Security Testing Guide
N/A	Not Applicable
NFC	Near Field Communication
OS	Operační systém

OVA	Open Virtualization Format
OWASP	The Open Web Appliacation Security Project
PDA	Personal Digital Assistant
PIN	Personal Identification Number
R	Resiliency
RAM	Random Access Memory
RCD	Remote Code Evaluation
RFD	Reflected File Download
SD	Secure Digital
SDK	Software Development Kit
SHA1	Secure Hash Algorithm 1
SSL	Secure Sockets Layer
SMS	Short Message Service
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

A Návod ke spuštění aplikace

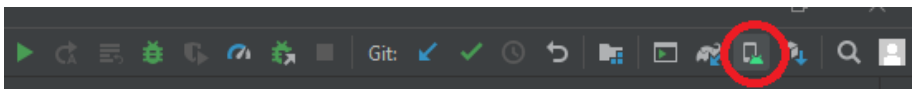
Tento návod popisuje jednotlivé kroky ke spuštění aplikace na emulátoru v prostředí Android Studio. Dále je zde uvedeno, jak si vytvořit uživatele a přihlásit se do aplikace.

Prerekvizity:

- nainstalované Android Studio
- nainstalovaný Oracle VM Virtual Box
- stažený virtuální obraz serveru PenterepMail (`https://www.penterep.com/penterepmail/download`)

Pokyny pro spuštění aplikace:

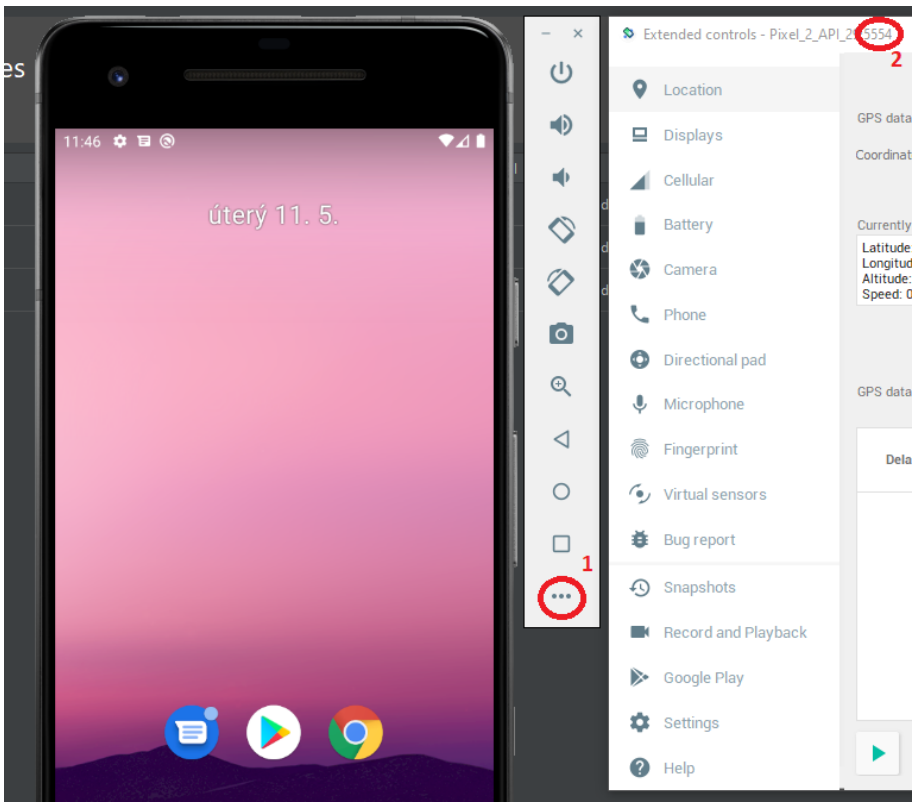
1. naimportujte virtuální obraz `PenterepMail (v1.2).ova` do Virtual Boxu
2. nastavte síťovou kartu jako Síťový most
3. spusťte virtuální obraz -> vyčkejte, dokud se neobjeví úvodní obrazovka s přidělenou IP adresou (pokud se adresa nepřidělí napoprvé, vypněte virtuální jednotku a opět ji spusťte)
4. do webového prohlížeče vypište získanou IP adresu -> po potvrzení se objeví webová aplikace PenterepMail
5. zaregistrujte se a poté se zkuste přihlásit do webové aplikace
6. spusťte Android Studio
7. spusťte emulátor (v případě, že nemáte zatím žádný emulátor k dispozici, postupujte podle dílčích kroků níže)
   (a) v Android Studiu klikněte vpravo nahoře na ikonu AVD Manager (obr. A.1)



Obr. A.1: Ikona AVD Managera v Android Studiu.

   (b) v zobrazeném okně klikněte na Create Virtual Device
   (c) vyberte položky Phone a Pixel 2
   (d) klikněte na Next
   (e) vyberte Release Name - Q (popřípadě klikněte u této položky na Download)
   (f) po výběru/instalaci kliněte na Next a dále pak na Finish
   (g) emulátor se vám zobrazí ve výběru
8. na řádku emulátoru klikněte na Launch (zelený trojúhleník)
9. vyčkejte, dokud emulátor nenaběhne (to může trvat několik minut)
10. přetáhněte soubor `PenterepMail.apk` na obrazovku emulátoru -> aplikace se začne instalovat

11. po instalaci klikněte na aplikaci PenterepMail v nabídce
12. povolte přístup k aplikaci
13. objeví se Aktivační obrazovka
14. zde zadejte jako telefonní číslo číslo portu, na kterém běží emulátor
    (a) klikněte na tři vodorovné tečky vpravo vedle emulátoru
    (b) zobrazí se nové okno pro nastavení emulátoru
    (c) nahoře v nadpisu okna (Extended controls) je za dvojtečkou zobrazený port - většinou se jedná o port 5554 (obr. A.2)



Obr. A.2: Zjištění portu emulátoru.

15. po zadání čísla vyčkejte, než se aplikace aktivuje
16. zobrazí se přihlašovací obrazovka, přes kterou se přihlásíte pomocí již registrovaných údajů

B Obsah elektronické přílohy

Obsahem elektronické přílohy je tato diplomová práce uložena ve formátu `.pdf`. Dále je součástí přílohy spustitelná aplikace pro mobilní zařízení se systémem Android s názvem `PenterepMail.apk`. Z důvodu velké velikosti souborů s kódy aplikace a s demonstračním videem, je součástí přílohy textový soubor, jehož obsahem je odkaz na sdílené úložiště, ve kterém se nachází jak kódy aplikace, tak demonstrační video.

- `/` ..... kořenový adresář elektronické přílohy
  - `PenterepMail.apk` ..... spustitelná aplikace
  - `Diplomová práce` ..... diplomová práce v pdf
  - `Odkaz na další přílohy.txt` ..... soubor s odkazem na další přílohy